НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем» спеціальності 121 Інженерія програмного забезпечення
на тему: «Програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data»

Виконала:
студентка IV курсу, групи КП-61
Корсакова Аліна Вадимівна _____

Керівник:
Старший викладач кафедри ПЗКС, к.ф.-м.н.,
Гречко А.В. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПСКС, к.т.н.
Бояринова Юлія Євгеніївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенці

Корсаковій Аліні Вадимівні

1. Тема проєкту «Програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data», керівник проєкту Гречко Анастасія Валеріївна, к.ф.-м.н., доцент, затверджені наказом по університету від «25» травня 2020 р. № 1181-с

2. Термін подання студенткою проєкту «13» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - обґрунтування теми дипломного проєкту;
   - обґрунтування вибору засобів реалізації;
   - розроблення програмного забезпечення;
   - аналіз розробленого програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - варіанти використання системи (креслення);
   - схема бази даних (креслення);
   - архітектура програмної платформи (плакат);
   - дерево проблем (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2019	
2.	Розроблення та узгодження технічного завдання	28.11.2019	
3.	Розроблення структури програмної платформи	15.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	20.02.2020	
7.	Програмна реалізація платформи	10.03.2020	
8.	Тестування програмної платформи	17.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	11.04.2020	
11.	Підготовка графічної частини дипломного проєкту	06.05.2020	
12.	Оформлення документації дипломного проєкту	01.06.2020	

Студентка Аліна КОРСАКОВА

Керівник проєкту Анастасія ГРЕЧКО

АНОТАЦІЯ

Даний дипломний проект присвячений створенню програмного забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data.

Проведено докладний аналіз наявних аналогів. Було проаналізовано необхідні функціональні та нефункціональні вимоги до розроблюваного продукту, виділено необхідні можливості, що має надавати система своїм користувачам.

Розроблена програмна система є веб-додатком, що містить динамічно оновлювані сторінки. У проєкті виконано порівняльний аналіз динаміки цінової політики автомобільного палива, а саме, вивчення зміни цін на різні види палива з 2016 року. Була вивчена екологічна складова проекту при переході на гібридні двигуни, дослідження обсягів викидів в атмосферу шкідливих речовин в залежності від виду палива. У системі передбачена реєстрація нових користувачів, а також обмеження доступу до певних можливостей вебдодатку незареєстрованого користувача.

Так само був виконаний аналіз обсягу продажів гібридних автомобілів за минулий період і прогноз розвитку цього сегмента в майбутньому.

ABSTRACT

The diploma project is devoted to the creation of a software system for analyzing the prospects of sales of hybrid cars using Big Data technology.

A detailed analysis of the existing analogues that create predictions of the price of digital currencies was carried out. Thanks to the found systems, the necessary functional and non-functional requirements for the software product were analyzed, and the necessary functionality of the system was identified.

The implemented software system is a web application that contains dynamically updated pages. In projects carried comparative analysis of automotive fuel pricing policy, namely the study of changes in prices of different fuels in 2016. The ecological component of the project during the transition to hybrid engines was studied, as well as the study of emissions of harmful substances into the atmosphere depending on the type of fuel. The system provides registration of new users, as well as restricting access to certain features of the web application of an unregistered user.

The analysis of sales of hybrid cars for the past period and the forecast of development of this segment in the future was also carried out.

ДП.045440-01-90 Програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмне забезпечення	5	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Технічне		
	завдання		
ДП.045440-03-81	Програмне забезпечення	56	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Пояснювальна		
	записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Програма та		
	Методика тестування		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-05-34	Програмне забезпечення	9	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Керівництво		
	користувача		
ДП.045440-06-99	Програмне забезпечення	1	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Архітектура		
	системи		
ДП.045440-07-99	Програмне забезпечення	1	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Схема бази		
	даних		
ДП.045440-08-98	Програмне забезпечення	1	
	для аналізу		
	перспективності збуту		
	гібридних автомобілів з		
	використанням технології		
	Big Data. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ПЕРСПЕКТИВНОСТІ ЗБУТУ ГІБРИДНИХ АВТОМОБІЛІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ BIG DATA

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Аліна КОРСАКОВА

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для аналізу збуту гібридних автомобілів з використанням технології Big Data.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для власників авто і потенційних ринкових покупців, які хочуть дізнатися про порівняльну статистику звичайних автомобілів з двигунами внутрішнього згоряння та гібридних автомобілів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Розроблена програмна система повинна забезпечувати наступні основні функції:

- можливість реєстрації та авторизації;
- можливість переглядати статистичні дані у вигляді таблиці та графіків;
- можливість сортувати вивід змін цін на паливо;
- можливість підписки на оновлення статистики;
- можливість змінювати свої персональні дані;
- можливість збереження графіків у вигляді PDF файлу;

Система парсингу даних з сайтів повинна бути розроблена на мові програмування Python з використанням фреймворку Scrapy. Бекенд сайту повинен бути розроблений на мові C#. Фронтенд сайту повинен бути розроблений на мові JavaScript та бібліотеки React.

Додаткові вимоги:

1) відображення на сторінці максимум 2 таблиць та 2 графіків одночасно;
2) наявність анімованих кнопок;
3) дизайн сторінок з використанням в якості базових білого та фіолетового кольорів.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Архітектура системи»;
   - «Схема бази даних».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2019

Розроблення та узгодження технічного завдання……....28.11.2019

Розроблення структури програмної платформи……...15.12.2019

Розроблення дизайну сторінок та графічних елементів……..03.02.2020

Програмна реалізація вебсайту…..……....10.03.2020

Тестування вебсайту…...…….....17.03.2020

Підготовка матеріалів текстової частини проєкту……..….11.04.2020

Підготовка матеріалів графічної частини проєкту……..…06.05.2020

Оформлення технічної документації проєкту……..…01.06.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ПЕРСПЕКТИВНОСТІ ЗБУТУ ГІБРИДНИХ АВТОМОБІЛЕЙ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ BIG DATA

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Аліна КОРСАКОВА

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ − програмне забезпечення

БД – база даних

UI – інтерфейс користувача

API (Application Programming Interface) – це інтерфейс програмування, інтерфейс створення додатків.

HTTP (англ. HyperText Transfer Protocol) – проткол передачі даних прикладного рівня. Спочатку використовувався переважно для HTML-документів, але зараз використовується для передачі даних довільного формату.

JSON (англ. JavaScript Object Notation) – формат даних, заснований на розширенні стандарту ECMA-262.

XML – (extensible Markup Linguae) – мова розмітки, що складеться з тегів та вмісту.

SQL (від англ. Structured query language) – декларативна мова програмування для взаємодії користувача з базами даних).

NoSQL (від англ. no only SQL) – база даних, що забезпечує інший механізм зберігання та видобування даних, ніж звичайний підхід таблицьвідношень в реляційних базах даних.

JS – скорочення від JavaScript.

Фреймворк – програмна платформа, що визначає структуру розроблюваної системи.

HTML (англ. *HyperText Markup Language*) – мова розмітки документів у мережі Інтернет. Інтерпретується браузером, а отриманий у результаті цієї інтерпретації відформатований текст відображається на екрані.

CSS (англ. *Cascading Style Sheets)* – формальна мова опису зовнішнього виду документів, написаних за допомогою мови розмітки (наприклад HTML).

CRUD – акронім з чотирьох основних дій, які можна виконати з базами даних, а саме: create (створення), read (читання), update (оновлення), delta(видалення).

Bootstrap – інструмент для розробки веб-сторінок, що містить готові шаблони кнопок, форм, блоків тощо.

СКБД – система керування базами даних.

ДВС – двигун внутрішнього згоряння.

ГСУ – гібридна силова установка.

ВСТУП

Перший гібридний автомобіль сучасної концепції запустила у виробництво компанія Toyota в грудні 1997 року. І якщо на кінець березня 2002 року компанією було продано близько 103 гібридних автомобілів, то в даний час загальні обсяги продажів автомобілів Toyota з гібридними двигунами перевищують 120000 автомобілів.

Аналіз перспектив збуту гібридних автомобілів в Україні на сьогодні є дуже актуальним питанням, яке цікавить як власників авто, так і потенційних ринкових покупців. Використання технології Big Data дозволяє провести якісний і повноцінний аналіз великого обсягу інформації для прийняття правильного рішення в сторону покупки гібридного автомобіля.

Одним з переваг гібридного автомобіля є зниження споживання палива. У той час, поки автомобіль знаходиться в пробці, він не генерує вихлопні з'єднання. Електродвигун дозволяє забезпечити моментальну зупинку і запуск.

Екологічна ефективність гібридів проявляється в русі на малих швидкостях, як відомо, що 60% сучасного забруднення повітря великих міст припадає на автотранспорт (пробки, перенасичений «тягучий» трафік на нижніх передачах), гібриди в такій ситуації справжня знахідка з мінімальним викидом шкідливих речовин. Сучасні моделі гібридів і зовсім використовують для руху з мінімальною швидкістю тільки електропривод, що означає повну відсутність викидів.

Метою даного дипломного проєкту є розроблення універсального та доступного програмного забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data. Розроблене програмне забезпечення має:

- Надавати розгорнуту інформацію про ціни на автомобілі та паливо для них у вигляді графіків.

- Надавати порівняльну статистику кількості шкідливих викидів у атмосферу автомобілями в залежності від витрати палива та його виду.
- Надавати інформацію щодо прогнозу розвитку сегмента гібридних автомобілів.
- Підтримувати відповідне шифрування даних задля підтримання інформаційної безпеки.

1. ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальні положення та огляд проблеми, яка вирішується ПЗ

Під гібридною силовою установкою автомобіля в даний час розуміють силовий агрегат, який об'єднує в одне ціле традиційний поршневий двигун внутрішнього згоряння, електрогенератор, акумулятор електроенергії, приводні електродвигуни та єдину систему мікропроцесорного керування.

Основними перевагами гібридних автомобілів, які сприяють їх поширенню:

- Автоматична підзарядка акумуляторної батареї. Гібридні автомобілі не вимагають самостійної підзарядки, так як вона відбувається автоматично за рахунок залишкової енергії двигуна. Робота двигуна задіяна в моменти активних навантажень і динамічного прискорення. Але в цей же час електрична установка бере на себе основні функції щодо забезпечення руху, а саме: старт машини, рух в міському циклі і підзарядка при гальмуванні.
- Прагнення до новітніх розробок у галузі автомобілебудування. Значна кількість майбутніх власників проявляє зацікавленість до найбільш сучасних автомобілів через бажання йти в ногу з часом.
- Екологічно чистий. Однією з найбільших переваг гібридного автомобіля перед бензиновим автомобілем є те, що він працює чистіше і має меньший показник витрати пального, що робить його екологічно чистим. Гібридний транспортний засіб працює з подвійним двигуном (бензиновий двигун та електродвигун), що скорочує споживання палива та економить енергію.

Проблеми, що перешкоджають поширенню гібридних автомобілів:

- Їх вартість. Високовольтна батарея, інвертора і електромотори-генератори силової установки збільшують вартість автомобіля на 30%.
- Ще одним істотним негативним чинником, як і для електроавтомобілів, є утилізація акумуляторних батарей, які рано чи пізно зношують ресурс перезаряджень.

Гібридизація автопарку України відбувається порівняно невеликими темпами. Частково, це пов'язано з поганою інформованістю про переваги гібридів і їх екологічної складової. Проте, розвиток безперечно присутній, тенденції до збільшення попиту на гібриди підтверджує більшість дистриб'юторів, так що можна бути впевненим, що частка екоавтомобілів на ринку України буде неухильно зростати.

З огляду на зацікавленість суспільства до збереження навколишнього середовища і високотехнологічних автомобільних продуктів, були сформовані проблеми області розробки дипломного проєкту. Головною проблемою проєкту є складність збору та аналізу великих обсягів інформації, а саме:

- відсутність порівняльної інформації екологічної складової гібридів та автомобілів з двигунами внутрішнього згорання;
- тривалий пошук цін на різне паливо за минулий період;
- консолідація інформації на одному вебресурсі.

Розроблюване програмне забезпечення у вигляді вебсайту спрямоване на вирішення проблем за допомогою впровадження перелічених вище заходів.

1.2. Аналіз аналогічних застосунків

Серед основних існуючих ресурсів можна виділити наступні: сайт Міністерства фінансів України, сайт УкрАвтопрому, сайт IRS Group та сайт Enerdata.

Сайт Міністерства фінансів України

Офіційний сайт Міністерства фінансів України інформує про зміни цін на автомобільне паливо, починаючи з 2016 року. Інформація представлена у вигляді таблиці та графіків під нею. В таблицю входять такі види палива: А 95+, А 95, А 92, ДП (дизельне паливо) та газ. Сайт містить зручну навігацію та багато корисної інформації.

Сайт УкрАвтопрому

Офіційний сайт УкрАвтопрому, на якому розміщена статистика продаж автомобілей по рокам. Проте, цю інформацію дуже складно знайти на самому сайті через відсутність структурованого представлення даних. Дизайн сайту застарілий та недосконалий. Відсутня наочна порівняльна статистика продаж у вигляді таблиць та графіків.

Сайт IRS Group

Сайт надає маркетингові послуги щодо аналізу ринку електро та гібридних автомобілів. Результати аналізу представлені у вигляді скріншотів, які показують кількість зареєстрованих легкових електромобілів та гібридів в Україні. Некоректне розміщення представлених скріншотів говорить про недостатнє опрацювання відображення об'єктів на сторінці.

Сайт Enerdata

Сайт показує обсяги викидів CO_2 у атмосферу, починаючи з 1990 року. Перевагою цього ресурсу є інтерактивна карта світу з переліком країн. В залежності від обсягу шкідливих викидів країни позначені різним відтінком синього кольору (чим більше, – тим темніше колір). На сайті реалізована можливість збереження статистики у вигляді файлу з розширенням .png. Проте, немає детальної інформації викидів CO_2 в залежності від марки і моделі автомобіля.

В результаті проведеного аналізу ринку аналогічних застосунків можна зробити висновок, що вищеперераховані ресурси надають окрему

інформацію, але вона не містить консолідованих висновків за потрібною темою дипломного проєкту.

1.3. Постановка завдання

За результатами проведених опитувань цільової аудиторії було отримано наступний список функціональних та не функціональних вимог, який має бути реалізований для вирішення основних проблем.

Система має три ролі користувачів: адміністратор, зареєстрований користувач та гість. Вона має ієрархічну модель ролей користувачів (вищий за ієрархію має можливості іншого): гість → зареєстрований користувач → адміністратор.

Гість повинен мати можливість:

- Переглядати статистичні дані у вигляді таблиці та графіків:
  - кількості шкідливих викидів у атмосферу в залежності від марки та моделі автомобіля;
  - ціни на автомобілі та зміни цін на автомобільне паливо за різні проміжки часу, починаючи з 2016 року;
  - прогноз розвитку сегмента гібридних автомобілів на основі обсягів продажів за попередні два роки.
- Сортувати вивід змін цін на паливо за роком.
- Сортувати вивід змін цін на автомобілі за виробником;
- Реєструватись у системі (тоді гість отримує роль зареєстрованого користувача).

Зареєстрований користувач повинен мати можливість:

- Підписатися на оновлення статистики – на задану користувачем поштову скриньку буде приходити інформація про оновлення статистичних даних на сайті.
- Змінювати свої персональні дані:
  - ім’я;
  - поштову скриньку;

    - логін;
    - пароль.
- Збереження графіків у вигляді PDF файлу:
    - порівняльна статистика кількості шкідливих викидів у атмосферу автомобілями;
    - зміна цін на автомобілі;
    - зміна цін на автомобільне паливо;
    - прогноз розвитку сегмента гібридних автомобілів.

Адміністратор повинен мати можливість:

- Модерувати користувачів – надати можливість відновити пароль від персонального кабінету користувача за зверненням на електронну скриньку адміністратора.
- Моніторинг своєчасного оновлення даних на сервері БД.
- Оновлювати порівняльну статистику кількості шкідливих викидів у атмосферу автомобілями за новими даними.
- Оновлювати інформацію щодо змін цін на автомобілі та автомобільне паливо за актуальністю.
- Оновлювати інформацію прогнозу розвитку сегмента гібридних автомобілів.

Вимоги до розробки системи:

- Система має бути сумісною з браузерами:
    - Google Chrome (від версіі 56.0.2924);
    - Mozilla Firefox (від версії 40.0);
    - Opera (від версії 27);
    - Safari (від версії 10).
- Система парсингу даних з сайтів повинна бути розроблена на мові програмування Python.
- Бекенд сайту повинен бути розроблений на мові C#.
- Сайт повинен бути розроблений на мові JavaScript.
- Дані повинні зберігатись у СКБД Microsoft SQL Server.

Вимоги до якості розроблюваного ПЗ:

- Відмовостійкість системи: одночасне підключення до серверу 100 клієнтами, максимальний час відновлення після відмови – не більше 2 годин.
- Зручний та ненавантажений для користувача інтерфейс: відображення на сторінці максимум 2 таблиць та 2 графіків одночасно.
- Використання бекендом не більше 450 МБ оперативної пам’яті.
- Масштабованість програмного забезпечення системи: гарантована стабільна робота сайту при додаванні не більше 10 сторінок.

2. ОБГРУНТУВАННЯ ВИБОРУ МЕТОДІВ АНАЛІЗУ

2.1. Обґрунтування вибору методу статистичного аналізу

Аналіз даних Big Data проходить три етапи: налаштування програмного аналітичного рішення (підключення її до всіх джерел інформації), вибору методології аналізу, засобів візуалізації (графіків, діаграм, таблиць), аналізу отриманої візуалізації і прогнозування подальших ситуацій на її основі [1].

Завдяки експоненціального зростання можливостей обчислювальної техніки, описаного в законі Мура, обсяг даних не може бути точним критерієм того, чи є вони великими. Наприклад, сьогодні великі дані вимірюються в терабайт, а завтра - в петабайт. Тому головною характеристикою Big Data є ступінь їх структурованості і варіантів представлення.

Визначальними характеристиками для великих даних є, крім їх фізичного обсягу, і інші, що підкреслюють складність завдання обробки і аналізу цих даних. Набір ознак VVV був вироблений компанією Meta Group в 2001 році з цілю визначити рівну значимість керування даними за наведеними аспектами.

Надалі з'явилися інтерпретації з чотирма V (додавалася veracity – достовірність), п'ятьма V (viability – життєздатність і value – цінність), сім'ю V (variability – мінливість і visualization – візуалізація). Але компанія IDC, наприклад, інтерпретує саме четвертий V як value (цінність), підкреслюючи економічну доцільність обробки великих обсягів даних у відповідних умовах.

Виходячи з вищенаведених визначень, основні принципи роботи з великими даними такі:

- Горизонтальна масштабованість. Кожного дня і кожну годину вебресурси наповнюються новою інформацією та збільшуються

в обсягах. Через це виникає потреба в обробці великих даних зі збереженням якості та часу обчислювань.

- Відмовостійкість. Цей принцип випливає з попереднього. Оскільки обчислювальних вузлів в кластері може бути багато (іноді десятки тисяч) і їх кількість, не виключено, буде збільшуватися, зростає і вірогідність несправності автомобіля. Необхідно визначити та виправити можливості несправної роботи.
- Методи роботи з великими даними повинні враховувати можливість таких ситуацій і передбачати превентивні заходи.
- Локальність даних. Для підвищення продуктивності обробки даних краще виконувати обчислення та збереження інформації на одному й тому самому обчислювальному пристрою.

Кореляційний аналіз

Кореляційний аналіз (correlation analysis) – статистичний метод вивчення взаємозв'язку між двома і більше випадковими величинами. У якості випадкових величин в емпіричних дослідженнях виступають значення змінних, вимірювані властивості досліджуваних об'єктів спостереження. Суть кореляційного аналізу полягає в розрахунку коефіцієнтів кореляції. Коефіцієнти кореляції можуть приймати, як правило, позитивні і негативні значення. В якості рівня залежності між змінними використовується коефіцієнт кореляції (r), який змінюється в межах від –1 до +1. Знак коефіцієнта кореляції дозволяє інтерпретувати напрямок зв'язку, а абсолютне значення – силу зв'язку.

Задачі кореляційного аналізу:

- Вимірювання рівня зв'язності (тісноти, сили, інтенсивності) двох і більше явищ.
- Відбір факторів, що роблять найбільш істотний вплив на результативну ознаку, на підставі вимірювання рівня зв'язності

між явищами. Істотні в даному аспекті фактори використовують далі в регресійному аналізі.

- Виявлення невідомих причинних зв'язків.

Популярність кореляційного аналізу пояснюється тим, що коефіцієнти кореляції відносно прості в розрахунку, і їх застосування не вимагає спеціальної математичної підготовки. З іншого боку – коефіцієнти кореляції легко інтерпретувати.

Однак кореляційний аналіз має свою специфіку і методику. Дуже важливо використання цього методу тільки при дотриманні передумов розрахунку того, чи іншого, коефіцієнта кореляції. Методика кореляційного аналізу припускає, не просто розрахунок коефіцієнтів кореляції, а й обов'язкову перевірку їх значущості, в основі якої лежить принцип перевірки статистичних гіпотез, побудова інтервальних оцінок коефіцієнтів кореляції.

Нерідкі випадки виникнення так званих «хибних кореляцій», що призводить до помилкових висновків. В цьому випадку при аналізі взаємозв'язку між кількісними змінними розраховують і аналізують окремі коефіцієнти кореляції.

Метод вибірки

Вибірковий метод (method of sampling) – статистичний метод дослідження загальних властивостей сукупності будь-яких об'єктів на основі вивчення властивостей лише частини цих об'єктів. Сукупність досліджуваних об'єктів, що цікавлять дослідника, називають генеральною сукупністю. Частина об'єктів, що підлягають вивченню, називають вибірковою сукупністю або вибіркою.

Вибірка або вибіркова сукупність – частина генеральної сукупності елементів, яка охоплюється експериментом (спостереженням, опитуванням).

Ключові питання вибіркового обстеження:

- кількісна характеристика вибірки або визначення мінімальної кількості спостережень (обсягу вибірки) для проведення дослідження;
- якісна характеристика вибірки або способи і методи формування вибіркової сукупності.

Головне завдання вибіркового дослідження – з мінімальним обсягом вибірки отримати найбільш точний опис цікавої генеральної сукупності на основі вибіркових даних. Досягти цього можна тільки на основі репрезентативної вибірки, тобто вибірка об'єктивно відображає властивості генеральної сукупності. Точність результатів вибіркових обстежень досягається за рахунок використання складних методів формування вибірки.

Мінімальний обсяг вибірки залежить від багатьох параметрів дослідження (оцінюваного показника або системи показників, способу і методів формування вибірки, варіації досліджуваних даних, заданої надійності одержуваних результатів, максимально допустимої помилки в оцінки показників) і визначається на основі формул математичної статистики або експертним шляхом.

Кластерний аналіз

Кластерний аналіз (cluster analysis) – сукупність багатомірних статистичних методів класифікації об'єктів за характерними ознаками, поділ сукупності об'єктів на однорідні групи, близькі за певними критеріями, виділення об'єктів певної групи.

Кластер – це групи об'єктів, виділені в результаті кластерного аналізу на основі заданої міри схожості або відмінностей між об'єктами. Об'єкт – це конкретні предмети дослідження, які необхідно класифікувати. Об'єктами при класифікації виступають, як правило, спостереження. Хоча можна проводити кластерний аналіз і по змінним. Класифікація об'єктів в багатовимірному кластерному аналізі відбувається за кількома ознаками

одночасно. Таким чином, основною цілю кластерного аналізу є виявлення груп подібних об'єктів у вибірці.

Сукупність багатовимірних статистичних методів кластерного аналізу можна розділити на ієрархічні методи і неієрархічні. Однак загальноприйнятої класифікації методів кластерного аналізу не існує, і до них відносять безліч алгоритмів машинного навчання, які вирішують задачу поділу сукупності на однорідні групи.

Сфера використання кластерного аналізу, через його універсальність, дуже широка. Кластерний аналіз застосовуються в різноманітних областях: економіці, маркетингу, соціології та інших.

Дисперсійний аналіз

Дисперсійний аналіз (ANOVA – analysis of variance) – статистичний метод вивчення взаємозв'язку. Застосовується для дослідження впливу однієї або декількох якісних змінних на одну залежну кількісну змінну. В основі дисперсійного аналізу лежить припущення, що одні змінні можуть розглядатися як причини (незалежні змінні), а інші як наслідки (залежні змінні або відгуки). Незалежні змінні в дисперсійному аналізі називають факторами, оскільки в ході експерименту дослідник може змінювати їх значення і аналізувати одержуваний результат залежною кількісної змінної.

Основна мета дисперсійного аналізу – дослідити значимість відмінності між середніми значеннями залежної кількісної змінної за групами фактору. Досягається це за допомогою розкладання загальної дисперсії залежної змінної на складові: дисперсію за рахунок розбиття на групи і дисперсію за рахунок інших факторів. Аналізуючи ці компоненти дисперсії можна оцінити частку впливу кожного фактору на залежну змінну. Окреме завдання дисперсійного аналізу – виявити, за рахунок яких саме груп йде відмінність середніх значень залежної змінної.

Розрізняють різні моделі дисперсійного аналізу. Залежно від числа факторів можуть бути однофакторні і багатофакторні дисперсійні моделі.

Залежно від числа спостережень в досліджуваних групах розрізняють збалансовані і незбалансовані моделі дисперсійного аналізу. Залежно від числа залежних змінних розрізняють одномірні і багатомірні моделі. В залежності від того залежні або незалежні вибірки, утворені категоріями фактора, виділяють дисперсійні моделі з незалежними вимірами та дисперсійний аналіз з повторними вимірами.

Факторний аналіз

Факторний аналіз (factor analysis) – багатомірний статистичний метод, що застосовується для вивчення взаємозв'язків між значеннями кількісних змінних. Основна ідея факторного аналізу полягає в тому, що наявні залежності між великим числом вихідних спостережуваних змінних визначаються існуванням набагато меншого числа прихованих або латентних змінних, які називаються чинниками.

Головними цілями факторного аналізу є: скорочення числа змінних і визначення структури взаємозв'язків між змінними. Тому факторний аналіз використовується або як метод скорочення даних або як метод класифікації. Факторний аналіз дозволяє досліднику ретельно описати об'єкт вимірювання з одного боку, враховуючи безліч вихідних тісно взаємопов'язаних між собою змінних, а з іншого боку компактно за допомогою невеликого числа змінних.

Проведення факторного аналізу передбачає володінням базисом статистичних знань: методами описового аналізу даних, методів перевірки статистичних гіпотез, знайомство з кореляційним і регресійним аналізом.

Факторний аналіз вперше виник і застосовувався для вимірювань в психології, але в даний час широко використовується для вирішення практичних завдань не тільки в психології, але і в соціології, політології, економіки, маркетингу, хімії та інших областях.

MapReduse

MapReduce – це модель розподілених обчислень від компанії Google, яка використовується в технологіях Big Data для паралельних обчислень

над дуже великими (до декількох петабайт) наборами даних в комп'ютерних кластерах, і фреймворк для обчислення розподілених задач на вузлах (node) кластера [2].

Авторами цієї обчислювальної моделі вважаються співробітники Google Джеффрі Дін (Jeffrey Dean) і Санджай Гемават (Sanjay Ghemawat), які взяли за основу дві процедури функціонального програмування: map, що застосовує потрібну функцію до кожного елементу списку, і reduce, яка об'єднує результати роботи map. У процесі обчислення безліч вхідних пар ключ/значення перетвориться в безліч вихідних пар ключ/значення.

MapReduce можна назвати головною технологією Big Data, тому що вона спочатку орієнтована на паралельні обчислення в розподілених кластерах. Суть MapReduce полягає в поділі інформаційного масиву на частини, паралельної обробки кожної частини на окремому вузлі і фінального об'єднання всіх результатів.

Технологія практично універсальна: вона може використовуватися для індексації вебконтенту, підрахунку слів у великому файлі, лічильників частоти звернень до заданої адреси, обчислення обсягу всіх вебсторінок з кожного URL-адреси конкретного хост-вузла, створення списку всіх адрес з необхідними даними і інших задач обробки величезних масивів розподіленої інформації. До галузей застосування MapReduce відносяться розподілений пошук і сортування даних, обробка статистики логів мережі, побудова інвертованих індексів, кластеризація документів, машинне навчання і статистичний машинний переклад.

Основні функції обчислювальної моделі:

- map – приймає на вхід список значень і якусь функцію, яку потім застосовує до кожного елементу списку і повертає новий список;
- reduse – перетворює список до єдиного атомарного значення за допомогою заданої функції, якій передають на кожній ітерації новий елемент списку і проміжний результат.

Для обробки даних відповідно до обчислювальної моделі MapReduce слід визначити обидві ці функції, вказати імена вхідних і вихідних файлів, а також параметри обробки.

Сама обчислювальна модель складається з трьох кроків комбінації вищенаведених функцій:

- Map – попередня обробка вхідних даних у вигляді великого списку значень. При цьому головний вузол кластера (master node) отримує цей список, ділить його на частини і передає робочим вузлам (worker node). Далі кожен робочий вузол застосовує функцію Map до локальних даних і записує результат в форматі «ключ-значення» в тимчасове сховище.
- Shuffle – робочі вузли перерозподіляють дані на основі ключів, які раніше створені функцією Map, таким чином, щоб всі дані одного ключа лежали на одному робочому вузлі.
- Reduce – паралельна обробка кожним робочим вузлом кожної групи даних по черзі проходження ключів і «склейка» результатів на master node. Головний вузол отримує проміжні відповіді від робочих вузлів і передає їх на вільні вузли для виконання наступного кроку. Одержаний після проходження всіх необхідних кроків результат – це і є рішення вихідної задачі.

У даному дипломному проєкті було використано MapReduse для обробки великих обсягів даних. Цей метод призначений для надвеликих об’ємів даних, але так як в його основі лежать розподілені обчислення це може значно прискорити обробку навіть даних середніх величин.

2.2. Обґрунтування вибору методу прогнозування

Метод прогнозування – метод, якій призначений для створення прогнозу шляхом детального вивчення об’єкта прогнозування.

Аналіз часових рядів

Аналіз часових рядів (time-series analysis) – сукупність статистичних методів для виявлення складових часового ряду і його прогнозування [3].

Часовий ряд або ряд динаміки – послідовність статистичних даних, зібраних в різні моменти часу, про значення яких-небудь параметрів досліджуваного процесу. Кожне значення часового ряду називається рівнем часового ряду. У часовому ряді кожному рівню має бути вказано час вимірювання або номер вимірювання по черзі. Методи аналізу часових рядів істотно відрізняються від методів аналізу даних простої вибірки. При аналізі часового ряду дослідника цікавлять не тільки статистичні характеристики часового ряду, але і враховується взаємозв'язок вимірювань з часом.

Основна мета аналізу часового ряду – побудувати прогноз його значень на майбутні періоди. Основні завдання аналізу часового ряду – зрозуміти, під впливом яких компонент формується значення часового ряду, і побудувати математичну модель для кожної компоненти або їх сукупності. Будь-який часовий ряд можна розкласти на такі складові: тренд, сезонну складову, циклічну складову і випадкову складову. Перші три компоненти утворюють невипадкову складову часового ряду. Випадкова складова присутня в будь-якому часовому ряді. Присутність компоненти невипадкової складової у структурі часового ряду не обов'язково.

Прогнозування тісно пов'язане з плануванням і використовується для ефективного прийняття рішень. Прогнозування може дати відповідь на питання: що найімовірніше очікувати в майбутньому щодо досліджуваного процесу або що необхідно зробити, щоб досягти заданого стану досліджуваного об'єкта прогнозування.

Метод ковзних середніх

Метод ковзних середніх – один з найпоширеніших методів згладжування часових рядів. Застосовуючи цей метод можна виключити

випадкові коливання та отримати результати, що показують вплив головних чинників.

Метод полягає у взаємному погашенні випадкових величин середніми величинами. Цей результат досягається заміною первинних рівнів часового ряду середнім арифметичним на обраному інтервалі часу. Розраховане значення відноситься до середини оброблюваного інтервалу часу. Далі інтервал зсувається на одне спостереження і обчислення повторюються для нового періоду. Слід зазначити, що періоди середньої є однаковими, що робить середню центрованою, тобто віднесеною до серединної точки інтервалу та являє собою рівень для цієї точки.

Згладжений ряд буде коротше початкого на $(n-1)$ спостережень, де n – показник інтервалу згладжування. Якщо значення n зростає, то коливання згладженого ряду суттєво знижується. Також істотно скорочується кількість спостережень, що створює труднощі. Слід зазначити що збільшення інтервалу згладжування дає більш плавний тренд.

Для вибору інтервалу згладжування слід керуватися цілями дослідження, а також періодом часу, у який відбувається процес, а значить і вплив випадкових факторів..

Даний метод використовується для короткострокового прогнозування. Його робоча формула:

$$y_{t+1} = m_{t-1} + \frac{1}{n} \cdot (y_t - y_{t-1}), \qquad (2.1)$$

де $t+1$ – прогнозований період; t – період, що передує прогнозованому періоду (рік, місяць і т.п.); y_{t+1} – прогнозований показник; m_{t-1}– змінна середня за два періоди до прогнозованого; n – кількість рівнів, що входять в інтервал згладжування; y_t – фактичне значення досліджуваного явища за попередній період; y_{t-1} – фактичне значення досліджуваного явища за два періоди, що передує прогнозованому.

Метод експоненціального згладжування

Експоненціальне згладжування - це сімейство методів прогнозування, яке обчислює середньозважене середнє за минулі спостереження як прогноз. Ваги спадають експоненціально, коли спостереження старіють. Як результат, чим недавніше спостереження, тим більша його вага в прогнозі. Сімейство експоненціальних методів згладжування моделює три аспекти часових рядів: рівень тенденції, нахил тренду та сезонний компонент. Ці три аспекти породжують три типи експоненціального згладжування: одинарне експоненціальне згладжування, подвійне експоненціальне згладжування та потрійне експоненціальне згладжування (також відоме як метод Холта-Вінтера).

Для середньострокового прогнозування доцільніше використовувати метод експоненціального згладжування. Його використовують для прогнозу на один строк вперед. Формула методу експоненціального згладжування:

$$U_{t+1} = a \cdot y_t + (1-a) \cdot U_t, \qquad (2.2)$$

де t – період, що передує прогнозованому; $t+1$ – прогнозований період; U_{t+1} – прогнозований показник; a – параметр згладжування; y_t – фактичне значення досліджуваного показника за період, що передує прогнозованому; U_t – експоненціально зважена середня для періоду, що передує прогнозованому.

Параметр згладжування α - це вибране число між нулем і одиницею, $0 < \alpha < 1$.

Експонентне згладжування дає уявлення про те, що найдавніші спостереження зазвичай дають найкраще уявлення про майбутнє, тому ми хочемо схему зважування із зменшенням ваг для старих спостережень. Вибір константи згладжування має важливе значення при визначенні експлуатаційних характеристик експоненціального згладжування. Чим менше значення α, тим повільніша відповідь. Більш великі значення α змушують згладжене значення швидко реагувати - не тільки нереалістичні

зміни, але й випадкові коливання. Проста модель експоненціального згладжування корисна лише для несезонних шаблонів, що мають приблизно нульову тенденцію, і для короткострокового прогнозування, оскільки, якщо ми продовжимо наступний період, прогнозоване значення для цього періоду має використовуватися як сурогат для фактичного попиту на будь-який прогноз наступного періоду. Отже, немає можливості додавати коригуючу інформацію (фактичний попит), і будь-яка помилка зростає експоненційно.

Однозначного методу для вибору оптимальної величини параметра згладжування a немає. В окремих випадках автор даного методу професор Браун пропонував визначати величину a, виходячи з довжини інтервалу згладжування. При цьому a обчислюється за формулою:

$$a = \frac{2}{n+1}, \tag{2.3}$$

де n – число спостережень, що входять в інтервал згладжування.

Задача вибору U_0 (експоненціально зваженого середнього початкового) вирішується наступними шляхами:

- якщо є дані про розвиток явища в минулому, то можна скористатися середньої арифметичною і прирівняти до неї U_0;
- якщо таких даних немає, то в якості U_0 і використовують вихідне перше значення бази прогнозу U_0.

Метод найменших квадратів

У процесі пошуку співвідношення двох змінних тенденція результатів оцінюється кількісно. Цей процес називають регресійним аналізом. Метод підгонки кривих – це підхід до регресійного аналізу. Цей метод підгонки рівнянь, який наближає криві до наведених необроблених даних, є найменшим квадратом.

Метод найменших квадратів регресії працює шляхом мінімізації суми квадрата помилки якомога менше, звідси і назва найменших квадратів. В основному відстань між лінією максимальної відповідності

умовам і помилки повинні бути зведені до мінімуму, наскільки це можливо. Це основна ідея методу регресії за методом найменших квадратів.

Перед застосуванням методу регресії найменших квадратів слід пам'ятати:

- Дані повинні бути без залишків, оскільки вони можуть призвести до упередженої та неправомірної лінії, що найкраще відповідає.
- Лінія, що найкраще підходить, може бути намальована ітеративно, поки ви отримано лінію з мінімальними можливими квадратами помилок.
- Цей метод добре працює навіть з нелінійними даними.
- Технічно різниця між фактичним значенням y та передбачуваним значенням y називається залишковим (позначає помилку).

Формула методу найменших квадратів:

$$y_{t+1} = a \cdot x + b, \tag{2.4}$$

де $t+1$ – прогнозований період; y_{t+1} – прогнозований показник; a і b – коефіцієнти; x – умовне позначення часу.

Розрахунок коефіцієнтів a і b здійснюється за такими формулами:

$$a = \frac{\sum_{i=1}^{n}\left(y_{ф} \cdot x\right) - \left(\sum_{i=1}^{n} x \cdot \sum_{i=1}^{n} y_{ф}\right)/n}{\sum_{i=1}^{n} x^2 - \left(\sum_{i=1}^{n} x\right)^2 / n}, \tag{2.5}$$

$$b = \frac{\sum_{i=1}^{n} y_{ф}}{n} - \frac{a \cdot \sum_{i=1}^{n} x}{n}, \tag{2.6}$$

де $y_{ф}$ – фактичні значення ряду динаміки; n – число рівнів часового ряду.

Для відображення закономірності розвитку досліджуваної події використовують згладжування часових рядів методом найменших

квадратів. Найскладнішими задачами передпрогнозного аналізу є: встановлення типи кривої та типу аналітичної залежності від часу.

Визначення виду функції, яка характеризує тренд за допомогою метода найменших квадратів, робиться шляхом створення ряду функції і порівняння їх між собою за формулою:

$$S = \sqrt{\frac{\sum_{i=1}^{n}\left(y_{ф} - y_{p}\right)^{2}}{n - p - 1}}, \qquad (2.7)$$

де $y_{ф}$ – фактичні значення ряду динаміки; y_{p} – розрахункові (згладжені) значення ряду динаміки; n – число рівнів часового ряду; p – число параметрів, що визначаються в формулах, що описують тренд (тенденцію розвитку).

Недоліки методу найменших квадратів:

- при спробі описати досліджуване економічне явище за допомогою математичного рівняння, точність прогнозу можливо для короткого періоду часу і рівняння регресії треба перераховувати по мірі надходження нової інформації;
- складність підбору рівняння регресії, яка є вирішуваною при використанні типових комп'ютерних програм.

Логістична регресія

Логістична регресія або логіт-регресія (logit model) – це статистична модель, яка використовується для прогнозування ймовірності появи події за допомогою логістичної функції.

Логістичну регресію відносять до моделей бінарного вибору. Регресійна модель бінарного вибору – це регресійна модель, в якій залежна змінна дихотомічна (бінарна). Значення факторів в моделях бінарного вибору повинні бути виміряні кількісною шкалою. Також в моделі бінарного вибору можна включати в якості чинників категоріальні змінні. Отже, в моделях бінарного вибору будується регресійна модель залежності

ймовірності того, що результативна дихотомічна змінна прийме значення 0 або 1 при заданому значенні факторів.

Для моделювання ймовірності дихотомічної залежної змінної вибирають спеціальну монотонно зростаючу функцію, яка може приймати значення в межах від 0 до 1.

В якості спеціальної функції в моделях бінарного вибору зазвичай використовують:

- логістичну функцію;
- функцію стандартного нормального розподілу.

В даному дипломному проєкті реалізований метод ковзних середніх. Метод ковзної середньої має ряд переваг перед іншими методами: він наочний при визначенні виду тренду і простий в тлумаченні ковзної середньої; ковзна середня визначає функцію тенденції, значення якої найбільш близькі до значень досліджуваного ряду, оскільки найкраща тенденція вибирається для окремих частин ряду (інтервалів згладжування); до досліджуваного ряду можуть бути додані нові рівні ряду; знаходження тенденції вимагає невеликої праці.

3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Обґрунтування вибору мови програмування

Для вибору оптимально необхідної мови програмування треба виконати аналіз існуючих та популярних мов. Далі потрібно шляхом їх порівняння визначити всі сильні та слабкі сторони, що дозволить обрати найбільш якісну для конкретного ПЗ.

Дипломну роботу можна розділити на 3 важливих етапи:

- парсинг даних з вебсторінок та наповнення бази даних;
- аналіз отриманих даних та побудування графіків;
- відображення результатів аналізу на вебсторінках.

Для реалізації поставлених цілей в найкоротші терміни часу було вирішено використовувати три мови програмування.

Огляд мови Python

Python – інтерпретована мова програмування високого рівня зі строгою типізацією [4]. Цю мову можна використовувати для розробки десктопних GUI застосунків, вебсайтів та вебдодатків. Неявна, але сувора типізація забезпечує менший обсяг коду для вирішення завдань, ніж в Java або C#. Прості правила синтаксису мови програмування додатково полегшують підтримку швидкої читабельності коду та коригування програми. Тому, можна виділити основні переваги Python:

- Python підтримується різноманітними операційними системами (Windows, Mac, Linux, Raspberry Pi тощо).
- Повністю підтримує об'єктно-орієнтоване та структуроване програмування.
- За допомогою мови Python розробники створюють програмні продукти з меншим об’ємом коду порівняно з деякими аналогічними мовами програмування.
- Має велику кількість надійних та корисних бібліотек.

Python широко використовується для розробки вебдодатків великіх масштабів. З огляду використання багатьох безкоштовних бібліотек та фреймворків для парсингу даних з вебсторінок було зроблено вибір у сторону Scrapy.

Scrapy – це безкоштовний фреймворк для швидкого вебсканування на високому рівні, який написаний на мові програмування Python. Він використовується для сканування вебсайтів та отримання з бажаних вебресурсів потрібної упорядкованої інформації. А також, використовується для великої кількості задач – від пошуку даних до моніторингу та автоматизованого тестування.

Ключові особливості Scrapy:

- Scrapy має вбудовану підтримку для отримання даних з джерел HTML з використанням виразів XPath і CSS.
- Це кросплатформна бібліотека (тобто написана на Python і працює на Linux, Windows, Mac і BSD) та легко розширювана.
- Швидше, ніж інші існуючі бібліотеки. Він може отримувати дані з сайтів у 20 разів швидше, ніж інші інструменти.
- Він споживає набагато менше пам'яті і ресурсів процесора.

Головним недоліком мови Python є низька швидкодія. Це зумовлено тим, що вона інтерпретована, тобто виконується за допомогою інтерпретатора рядок за рядком замість компілятора. Проте, даний етап не вимагає високої швидкості виконання, а тому, з огляду на простоту, Pyhton є досить ефективним.

Огляд мови C#

C# – сучасна, об'єктно-орієнтована і строго типізована мова програмування. C# відноситься до відомого сімейства мов C, тому синтаксис схожий на Java або C ++ [5].

C# підтримує спадкування, статичну типізацію, поліморфізм та перевантаження операторів. Для побудови високоструктурної та динамічної архітектури додатку розробниками надається перевага

об'єктно-орієнтованого підходу. Ця мова з кожним оновленням стає все краще та виникають нові функціональні можливості, а саме: лямбда, динамічне зв'язування, асинхронні методи і т.п.

Переваги C#:

- Компіляція та час виконання мови C# досить швидкі.
- У програмуванні на C# встановлена дуже ефективна система, яка автоматично збирає та видаляє сміття.
- Для пошуку та ліквідації помилок використовується обробка винятків.

.NET Core – програмна платформа, випущена компанією Microsoft в 2016 році для розробки програмного забезпечення з відкритим вихідним кодом [6]. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR).

Переваги .NET Core:

- Підтримка декількох мов. Common Language Runtime (CLR) – основа платформи, за рахунок чого .NET Core забезпечує функціонування декілька мов програмування. При компіляції код на будь-якому з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language). Таким чином виникає можливість розробки на різних мовах програмних застосунків.
- Бібліотека класів .NET Core поєднує у собі підтримування для різних мов програмування.
- Багатогранність методик. Бібліотека класів .NET Core та CLR – основа для різноманітного поєднання технологій, які використовуються на практиці для розробки додатків.

Головним недоліком C# є орієнтованість, в основному, тільки на .NET Core (на Windows платформу). Для розробки комерційного програмного забезпечення необхідна недешева офіційна ліцензія Microsoft

Visual Studio, проте для студентів існує безкоштовна версія. З огляду на всі переваги .NET Core, бекенд системи реалізований на мові C#.

Огляд мови JavaScript

JavaScript – це кросплатформна, об'єктно-орієнтована мова, яка використовується для створення інтерактивної вебсторінки (зі складною анімацією, кнопками, що можна натискати, спливаючими меню тощо). Існують також більш просунуті серверні версії JavaScript, такі як Node.js, які дозволяють додати більше функціональності на вебсайт, ніж просто завантажувати файли. Вона не надає низькорівневого доступ до пам'яті або процесора, оскільки вона була створена для браузерів, які цього не потребують.

Переваги JavaScript:

- JavaScript, як правило, дуже швидкий, оскільки він може виконуватись відразу в браузері клієнта при необхідності. Поки він не вимагає сторонніх ресурсів, JavaScript не сповільнюється викликами на сервер.
- Дана мова використовується у багатьох розробках та добре поєднується з іншими мовами програмування.
- Дає можливість створювати багатофункціональні інтерфейси.

Недоліки JavaScript:

- Оскільки код виконується на комп'ютері користувачів, в деяких випадках його можна використовувати в зловмисних цілях.
- Браузери іноді інтерпретують JavaScript по-різному. Це ускладнює написання кросбраузерного коду.

Не зважаючи на перераховані вище недоліки, відображення результатів аналізу на вебсторінці було вирішено розробити на мові JavaScript.

3.2. Обґрунтування вибору системи керування базою даних

Система керування базами даних (СКБД) – це система для побудови нової бази даних, також використовується для наповнення інформацією, для коригування вмісту та представлення результатів. Найпопулярніші СКБД: MySQL, PostgreSQL, Microsoft SQL Server, MongoDB.

Огляд СКБД MySQL

MySQL підтримує кросплатформність (Linux, UNIX та Windows). Дану СКБД використовують у різних застосунках, MySQL найчастіше асоціюється з вебдодатками та публікаціями в Інтернеті.

MySQL використовує модель клієнт-сервер. Ядро MySQL – сервер MySQL, призначений для обробки команд бази даних. MySQL-сервер застосовується у вигляді окремої програми для подальшого використання моделі клієнт-серверу, а також у вигляді бібліотеки для вбудування в програму.

Дана СКБД використовується для швидкісного аналізу великих об’ємів інформації. Завдяки використанню MySQL розробники мають можливість надсилати бази даних на різні сервери.

Переваги MySql:

- підтримка різних типів таблиць (MyISAM, InnoDB, EXAMPLE і ін.);
- економне споживання ресурсів;
- синхронізація з іншими базами даних (Oracle, DB2 та ін.).

Недоліки MySQL:

- фрагментарне використання SQL (якщо раніше був досвіт роботи зі стандартною мовою, при впровадженні СКБД можуть виникнути проблеми);
- платна техпідтримка (навіть для безкоштовних версій).

Огляд СКБД PostgreSQL

PostgreSQL – безкоштовна об'єктно-реляційна система керування базами даних [8]. Ця система включає в себе багато функції, які

допомагають користувачам під час розробки ПЗ і адміністраторам для безпечного збереження інформації та планування архітектури. Така різноманітність функціональних можливостей PostgreSQL якісно підтримує керування даними незалежно від їх кількості.

Масштабна об'єктно-реляційна база даних, яка працює на Linux, Windows, OSX і деяких інших системах. У PostgreSQL 10 є такі функції, як логічна реплікація, декларативне розбиття таблиць, поліпшені паралельні запити, більш безпечна аутентифікація по паролю на основі SCRAM-SHA-256.

Переваги PostgreSQL:

- висока масштабованість;
- можливість налаштування власного інтерфейсу;
- універсальність (підходить для використання на більшості популярних платформ).

Недоліки PostgreSQL:

- підвищена витрата ресурсів;
- проблеми з хостингом.

Огляд СКБД Microsoft SQL Server

Microsoft SQL Server - це система керування реляційними базами даних від Microsoft. Також є однією з найбільш популярних СКБД в світі. Використовується для різноманітних програмних продуктів: від невеликих додатків до великих високонавантажених проєктів. Працює як з локальними, так і з хмарними серверами (їх навіть можна використовувати одночасно). З 2016-го року сумісна з Linux, але оптимально використовувати її в операційних системах сімейства Windows.

Переваги Microsoft SQL Server:

- синхронізація з іншими програмними продуктами Microsoft;
- високий захист даних (шифрування, динамічне маскування та ін.);

- простий інтерфейс, завдяки чому з цією СКБД відносно легко працювати і вести адміністрування;
- відмінна масштабованість.

Недоліки Microsoft SQL Server:

- підвищене споживання ресурсів;
- висока ціна за ліцензію.

Огляд СКБД MongoDB

MongoDB використовує документо-орієнтовану модель даних. Це означає, що вона зберігає інформацію у вигляді документів, а не таблиць. Документи вміщають в себе багатокомпонентну інформацію та являють собою сховище пар ключ/значення. Дана СКБД працює швидше інших реляційних баз даних, краще масштабована та легка у використанні, через це користується популярністю серед розробників [7].

MongoDB підтримує кросплатформність (Windows, Linux, MacOS, Solaris), тому що розроблена на базі мови програмування C++.

Переваги MongoDB:

- можливість роботи з будь-якими видами даних;
- автоматична фрагментація;
- висока продуктивність;
- потужна горизонтальна масштабованість.

Недоліки:

- для роботи з реляційними базами доведеться вручну переписати код;
- дуже нестійка до різного роду атакам;
- для комерційного використання потрібна платна версія.

Серед вищеперерахованих СКБД було обрано Microsoft SQL Server через його надійність та сумісність з мовою програмування C#.

3.3. Загальний опис системи

Даний дипломний проєкт має бути розроблений у вигляді вебсайту, щоб надавати користувачам інформацію щодо аналізу перспективності збуту гібридних автомобілів на ринку України. Система має підтримувати розділення на ієрархічну модель ролей користувачів.

У системі передбачені 3 види ролей користувача:

- гість;
- зареєстрований користувач;
- адміністратор.

Гість

Може переглядати порівняльну статистику кількості шкідливих викидів у атмосферу автомобілями, прогноз розвитку сегмента гібридних автомобілів, інформацію щодо змін цін на автомобілі та паливо. Має доступ до сортування виводу змін цін на паливо за роком, а також сортувати вивід цін на автомобілі за виробником. Може зареєструватись у системі і тоді він отримує роль зареєстрованого користувача.

Зареєстрований користувач

Має можливість виконувати такі ж самі функції як і гість. Має можливість переглядати порівняльну статистику кількості шкідливих викидів у атмосферу автомобілями, прогноз розвитку сегмента гібридів, інформацію цін на автомобілі та паливо. Має можливість оновлювати свої персональні дані, а саме: ім’я, поштова скринька, логін та пароль. Може зберігати графіки у вигляді PDF файлу, а саме: графік порівняльної статистики кількості шкідливих викидів у атмосферу автомобілями, графік цін на автомобілі та зміни цін на пальне, графік прогнозу розвитку сегмента гібридних автомобілів. Має можливість підписки на оновлення статистики, після чого на вказану при реєстрації поштову скриньку буде приходити інформація про оновлення даних на сайті.

Адміністратор

Виконує роботу по оновленню контенту на вебсторінках, а саме: порівняльна статистика кількості шкідливих викидів у атмосферу автомобілями, інформація цін на автомобілі та зміни цін на паливо, прогноз розвитку сегмента гібридних автомобілів. Має доступ до деякої особистої інформації користувачів, а саме: ім’я користувача та поштова скринька. Має можливість модерувати користувачів та надавати можливість відновити пароль від персонального кабінету користувача, на його поштову скриньку буде приходити лист з підтвердженням, після чого користувач має відновити свій пароль.

На початку роботи із системою користувач потрапляє на головну сторінку. Після чого він має можливість зареєструватись, натиснувши на кнопку «Registration», яка знаходиться на хедері сторінки. Його перенаправить на сторінку реєстрації, де потрібно ввести у відповідні поля ім’я користувача, поштову скриньку, логін та пароль. Якщо дані вірні, то користувач буде зареєстрований у системі, якщо ні – отримує помилку з інформацією. Якщо користувач вже зареєстрований, то він може увійти у систему. Для цього потрібно натиснути на кнопку «Log In», яка знаходиться на хедері сторінки. Користувача перенаправить на сторінку входу, де потрібно ввести свій логін та пароль. Якщо користувач вже є у системі та пройде перевірку у БД, то операція входу пройде успішно, якщо ні – отримує помилку з інформацією. Після успішного входу у систему користувач потрапляє на головну сторінку. Щоб увійти у систему як адміністратор, потрібно на сторінці «входу» у систему ввести логін та пароль, які завчасно передбачені адміністрацією сайту.

З головної сторінки зареєстрований користувач може потрапити на такі сторінки відповідно:

- Profile – сторінка особистого кабінету користувача, де він може переглянути свою інформацію, змінити за потребою та підписатися на оновлення статистики.

- Admin – сторінка особистого кабінету адміністратора, з’являється якщо користувач успішно увійшов у систему як адміністратор. Після чого може розпочати роботу з модерування сайту та оновлення контенту на сторінках.
- Pollutions – сторінка, на якій буде відображена інформація з порівняльною статистикою кількості шкідливих викидів у атмосферу автомобілями, в залежності від виду палива.
- CarPrice – сторінка, на якій буде відображена інформація з цінами на автомобілі.
- CarSales – сторінка, на якій буде статистика продажів автомобілів, починаючи з 2016 року. Також на цій сторінці можна сортувати статистику продажів за роком.
- FuelSales – сторінка, на якій буде відображена інформація зміни цін на автомобільне паливо. Також на цій сторінці можна сортувати вивід зміни цін на паливо за роком.
- Forecast – сторінка, на якій буде відображена інформація прогнозу кількості зареєстрованих гібридних автомобілів в Україні.

Щоб вийти із системи потрібно натиснути на кнопку «Log Out», яка знаходиться на хедері сторінки. Після чого зареєстрований користувач або адміністратор отримує роль гостя та перенаправляється на головну сторінку. Щоб продовжити роботу з додатком як зареєстрований користувач або адміністратор потрібно заново виконати вхід у систему.

3.4. Архітектура системи

Система розділена на три частини:

- Парсинг даних з вебсторінок та заповнення БД.
- Обробка отриманих даних та створення статистики.
- Відображення статистики на вебсторінках.

Для першої частини було вирішено використовувати мову програмування Pyhton та безкоштовний фреймворк Scrapy. Спочатку потрібно було вибрати джерела, з яких потрібно збирати інформацію. Потім був написаний скрипт, який парсив потрібні дані з вебсторніки. Після чого, зберігав отримані дані у таблицу у БД.

Для другої частини було вирішено використовувати мову програмування C#. На цій мові був написаний бекенд системи з використанням фреймворку ASP.NET Core Web API. Архітектура цієї частини використовує шаблон MVC [9]. Концепція шаблону MVC передбачає розділення додатка на три компоненти:

- Model (модель) – характеризує дані та пов’язану з ними логіку.
- View (представлення) – відповідає за візуальну частину або призначений для користувача інтерфейс, зазвичай html-сторінка, через який користувач взаємодіє з додатком.
- Controller (контролер) – це центральний компонент MVC, за допомогою якого забезпечується зв'язок між розробником та ПЗ, представленням і сховищем даних.

В цій частині реалізовані алгоритми вибору статистичних даних для відображення на сторінці, а також валідація вхідних параметрів запитів на отримання інформації.

Для третьої частини було вирішено використовувати мову програмування JavaScript та відкритий фреймворк React [10]. В цій частині реалізована подієва модель, яка дозволяє відправляти запити користувачів на бекенд системи для отримання інформації та відображення її на вебсторінці. Архітектуру системи наведено на рис. 3.

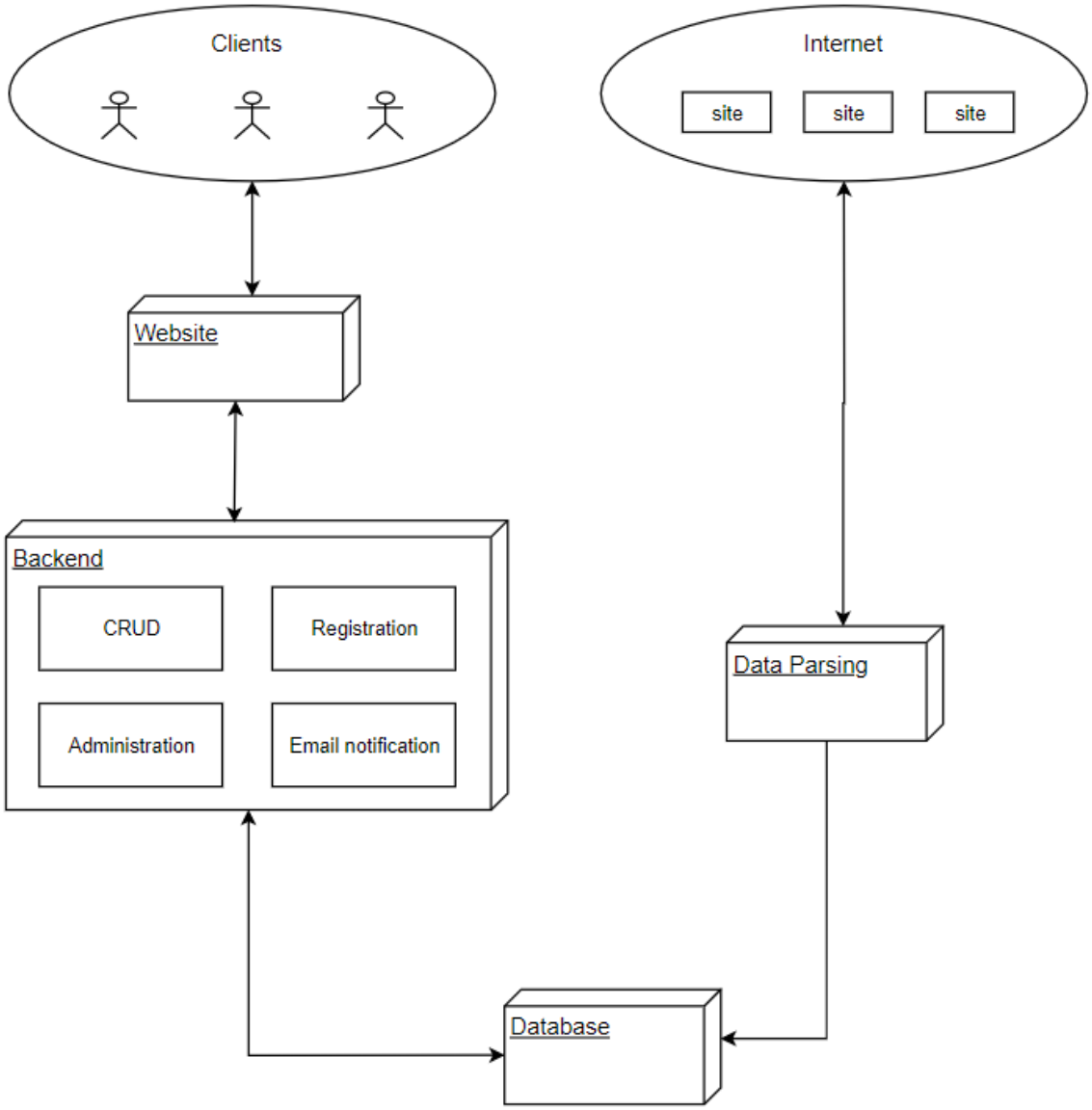


Рис. 3. Архітектура системи

Взаємодія із СКБД

Для даного дипломного проєкту була обрана MS SQL Server як СКБД [11]. Щоб налагодити взаємодію мови програмування Python та SQL Server було використано бібліотеку PyODBC. Цю бібліотеку можна використовувати як для Windows, так і Linux. Пакет PyOBDC реалізує специфікацію DB API 2.0.

Схема бази даних

Для реалізації системи було створено 8 таблиць:

- ListCarModel;

- ListFuelType;
- ListPollution;
- ListStandartType;
- Car;
- CarSales;
- Pollution;
- FuelSales;
- User.

На рис. 4 зображено розроблену схему бази даних.

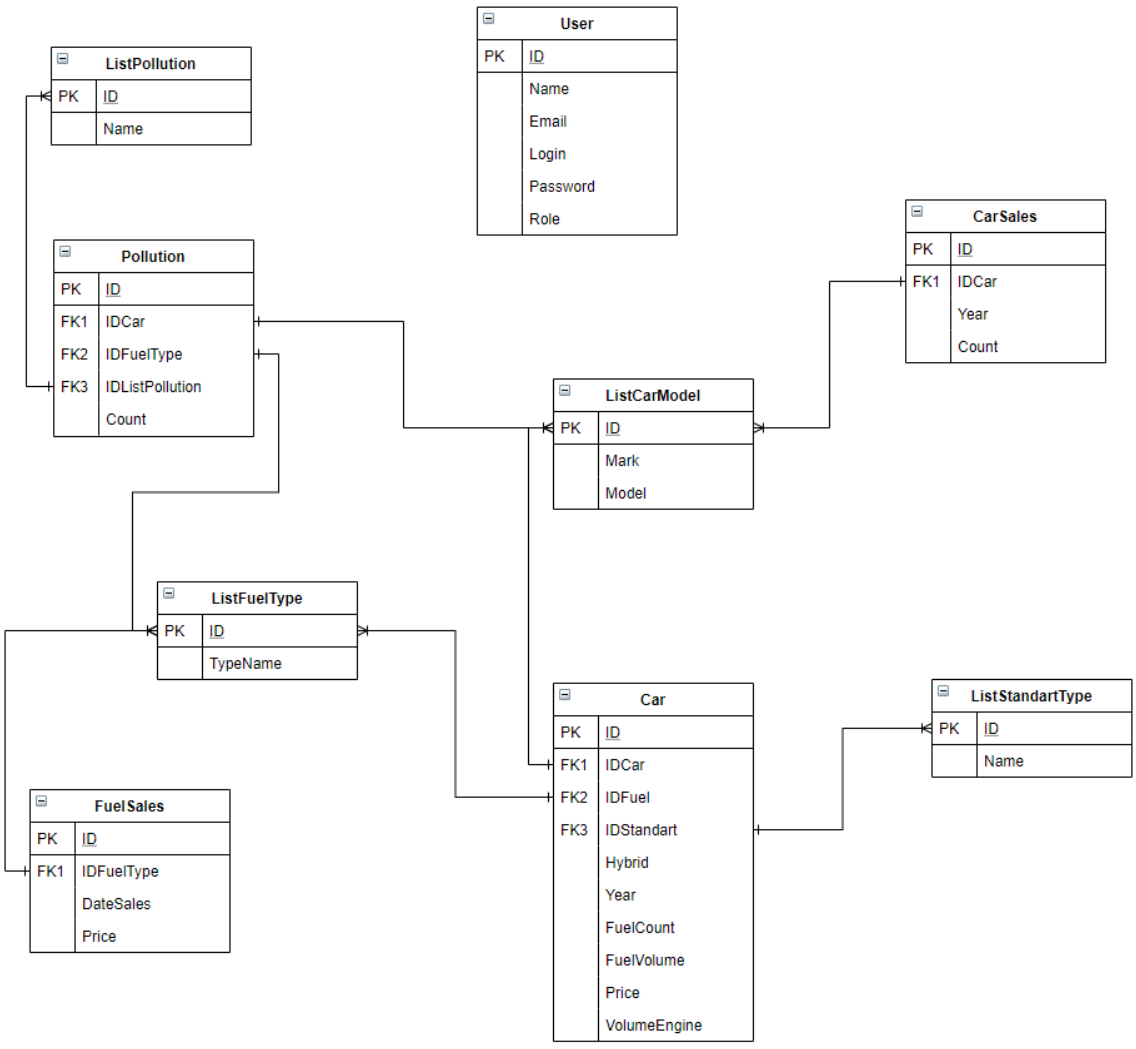


Рис. 4. Схема бази даних

ListCarModel

Таблиця, яка містіть список моделей і марок автомобілів. Назви полів та їх типів зазначені нижче у таблиці 1.

Таблиця 1

Поля таблиці ListCarModel

Поле	Тип даних
ID	Int AUTOINCREMENT
Mark	Text
Model	Text

ListFuelType

Таблиця, яка містить список видів палива. Назви полів та їх типів зазначені нижче у таблиці 2.

Таблиця 2

Поля таблиці ListFuelType

Поле	Тип даних
ID	Int AUTOINCREMENT
TypeName	Text

ListPollution

Таблиця, яка містить список видів забруднення, які викидають автомобілі. Назви полів та їх типів зазначені нижче у таблиці 3.

Таблиця 3

Поля таблиці ListPollution

Поле	Тип даних
ID	Int AUTOINCREMENT
Name	Text

ListStandartType

Таблиця, яка містить список екологічних стандартів автомобілів. Назви полів та їх типів зазначені нижче у таблиці 4.

Таблиця 4

Поля таблиці ListStandartType

Поле	Тип даних
ID	Int AUTOINCREMENT
Name	Text

Car

Таблиця, яка містить список автомобілів та детальну інформацію про них. Назви полів та їх типів зазначені нижче у таблиці 5.

Таблиця 5

Поля таблиці Car

Поле	Тип даних
ID	Int AUTOINCREMENT
IDCar	Int (Foreign Key of ListCarModel)
IDFuel	Int (Foreign Key of ListFuelType)
Hybrid	Boolean

Продовження табл. 5

Year	Int
FuelCount	Float
FuelVolume	Int
IDStandart	Int (Foreign Key of ListStandartType)
Price	Float
VolumeEngine	Int

CarSales

Таблиця, яка містить список продаж окремого автомобіля. Назви полів та їх типів зазначені нижче у таблиці 6.

Таблиця 6

Поля таблиці CarSales

Поле	Тип даних
ID	Int AUTOINCREMENT
IDCar	Int (Foreign Key of Car)
Year	Int
Count	Int

Pollution

Таблиця, яка містить інформацію про окремий тип забруднення від автомобіля. Назви полів та їх типів зазначені нижче у таблиці 7.

Таблиця 7

Поля таблиці Pollution

Поле	Тип даних
ID	Int AUTOINCREMENT
IDCar	Int (Foreign Key of Car)
IDFuelType	Int (Foreign Key of ListFuelType)
IDListPollution	Int (Foreign Key of ListPollution)
Count	Float

FuelSales

Таблиця, яка містить інформацію про продаж різних типів пального для автомобілів. Назви полів та їх типів зазначені нижче у таблиці 8.

Таблиця 8

Поля таблиці FuelSales

Поле	Тип даних
ID	Int AUTOINCREMENT
IDFuelType	Int (Foreign Key of ListFuelType)
DateSales	Text
Price	Float

User

Таблиця, яка містить інформацію про користувача системи. Назви полів та їх типів зазначені нижче у таблиці 9.

Таблиця 9

Поля таблиці User

Поле	Тип даних
ID	Int AUTOINCREMENT
Name	Text
Email	Text
Login	Text
Password	Text
Role	Text

3.5. Вимоги до безпеки системи

Невід'ємною частиною створення якісного програмного продукту є оцінка ризиків. Управління ризиками проєкту – це страховка, за допомогою якої можна вчасно врятувати важливу складову проєкту, будь то гроші, час або навіть рівень якості продукту. До того ж, профілактичні заходи часто виходять дешевше і швидше вирішення виниклих проблем.

Головна мета роботи з ризиками – вибрати і застосувати вірну стратегію управління. Які ризики проєкту як краще вирішувати, підкаже ретельний аналіз. Для кожного ризику можна підібрати одну стратегію або скомбінувати кілька. В результаті повинна бути готова основна стратегія і на випадок неефективності основний – резервна.

Під час роботи були розглянуті актуальні уразливості та загрози безпеці вебдодатку та даних. Серед основних ризиків, які можуть виникнути під час роботи системи, можна виділити:

- SQL-ін’єкції коду.
- Неправильне розмежування прав доступу.
- Недостатній контроль вхідних даних.
- Погане управління паролями та персональними даними.

- Неконтрольоване використання ресурсів.
- Відсутність або проведення в недостатньому обсязі тестування програмного продукту.

Результати ідентифікування ризику безпеки програмного забезпечення зазначені нижче у таблиці 10.

Таблиця 10

Результати ідентифікування ризику безпеки програмного забезпечення

№ з/с	Уразливість	Загроза	Наслідки
1	SQL-ін’єкції коду	Надають можливість атакуючому виконати довільний запит до бази даних	Можливість втрати даних
2	Неправильне розмежування прав доступу	Може призвести до того, що користувач отримає доступ до чужих даних	Порушення конфіденційності даних
3	Недостатній контроль вхідних даних	Атакуючий може скласти спеціальний тимчасовий запрос і система відреагує на нього, запропонувати підвищені привілегії або доступні критично важливим даним	Втрата даних
4	Погане управління паролями та персональними даними	Може призвести до привласнення чужого призначеного для користувача ідентифікатора	Втрата конфіденційних даних

Продовження табл. 10

5	Неконтрольоване використання ресурсів	Призводить до того, що споживання пам'яті програмою неконтрольовано зростає	Може бути аварійна зупинка
6	Відсутність або проведення в недостатньому обсязі тестування ПЗ	Неможливо оцінити та усунути всі ризики	Можуть виникнути непередбачені помилки

Визначення вимог до безпеки системи

Після визначення ризиків проєкту були сформовані вимоги забезпечення безпеки програм та даних. Завдяки чому ризики будуть усунені.

- Забезпечити захист системи від SQL-ін’єкцій.
- Протестувати систему на наявність збоїв роботи та помилок.
- Вести логування станів сервера та станів клієнта, які призводять до помилок.
- Контролювати всі вхідні дані, дозволяючи лише свідомо правильні.

3.6. Особливості реалізації

У дипломному проєкті було використано шаблон MVC.

Клас ListCarModel

Відповідає за взаємодію із таблицею у БД, яка містить список назв моделей та марок автомобілів. Включає у собі такі поля:

- ID – унікальний ідентифікатор у БД;
- mark – назва марки автомобіля;
- model – назва моделі автомобіля.

Клас ListFuelType

Відповідає за взаємодію із таблицею у БД типів палива. Включає у собі такі поля:

- id – унікальний ідентифікатор у БД;
- typeName – назва виду палива.

Клас ListPollution

Відповідає за взаємодію із таблицею у БД типів забруднення:

- id – унікальний ідентифікатор у БД;
- name – назва виду забруднення.

Клас ListStandartType

Відповідає за взаємодію із таблицею у БД типів екологічних стандартів автомобілів. Включає у собі такі поля:

- id – унікальний ідентифікатор у БД;
- name – назва екологічного стандарту.

Клас Car

Відповідає за взаємодію із таблицею у БД автомобілів. Містить такі поля:

- id – унікальний ідентифікатор у БД;
- idCar – ідентифікатор із таблиці ListCarModel;
- idFuel – ідентифікатор із таблиці ListFuelType;
- hybrid – приймає значення true, якщо автомобіль гібридний, якщо зі звичайним двигуном внутрішнього горіння – приймає значення false;
- year – рік випуску машини;
- fuelCount – витрати пального, кількість літрів на 100 км пробігу;
- fuelVolume – об'єм паливного бака;
- idStandart – ідентифікатор із таблиці ListStandartType;
- price – ціна автомобіля;
- volumeEngine – потужність двигуна автомобіля.

Клас CarSales

Відповідає за взаємодію із таблицею у БД, яка містить кількість продажів автомобілів. Включає у собі такі поля:

- id – унікальний ідентифікатор у БД;
- idCar – ідентифікатор із таблиці Car;
- year – рік, в якому була здійснена покупка;
- count – кількість проданих машин.

Клас FuelSales

Відповідає за взаємодію із таблицею у БД, яка містить ціни на паливо. Включає у собі такі поля:

- id – унікальний ідентифікатор;
- idFuelType – ідентифікатор із таблиці ListFuelType;
- dateSale – дата продажу палива;
- price – ціна на паливо.

Клас Pollution

Відповідає за взаємодію із таблицею у БД забруднень. Включає у собі такі поля:

- id – унікальний ідентифікатор;
- idCar – ідентифікатор із таблиці ListCar;
- idFuelType – ідентифікатор із таблиці ListFuelType;
- idListPollution – ідентифікатор із таблиці ListPollution;
- count – кількість забруднень.

Клас User

Відповідає за взаємодію із таблицею у БД користувачів. Включає у собі такі поля:

- id – унікальний ідентифікатор;
- name – ім’я користувача;
- email – поштова скринька користувача;
- login – логін;

- password – пароль користувача, який захешований;
- role – роль користувача, може приймати значення «admin» або «registered».

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Аналіз реалізованої системи

У результаті розробленого вебдодатку було створено ПЗ, призначене для аналізу перспективності збуду гібридних автомобілів.

Відповідно до сформованих функціональних вимог до системи (підрозділ 1.3. «Загальні вимоги до системи») було розроблено відповідні рішення:

- У системі було реалізовано ієрархічна модель ролей користувача: гість – зареєстрований користувач – адміністратор. Різні рівні доступу користувачів забезпечується відповідним поле у БД, де вказується значення «admin», «registered». Дані гостя не зберігаються у БД.
- Система має персональний кабінет користувача. Якщо користувач має роль «registered», то на сторінці «Profile» він може змінити свої особисті дані та підписатися на оновлення статистики, якщо користувач «admin» – зі сторінки профілю він може виконувати роботу з модерування сайту, а саме: блокувати деяких зареєстрованих користувачів. Підписка на оновлення статистики було реалізовано за допомогою класу «Subscription». Також, користувач має можливість збереження графіків у вигляді PDF файлу. Розподілення можливостей користувача було реалізовано за допомогою класу «User».
- Сторінка з порівняльною статистикою кількості шкідливих викидів речовин у атмосферу автомобілями було реалізована за допомогою класу «Pollution». Інформація на сайті відображена у вигляді графіків.
- Сторінка з інформацією щодо змін цін на автомобілі та паливо була реалізована завдяки класам «CarSales» та «FuelSales». Інформація на сайті відображена у вигляді графіків за окремий

проміжок часу. Користувачі системою можуть відсортувати вивід даних за існуючими фільтрами. Сортувати вивід змін цін на паливо за видом палива та роком. Сортувати вивід змін цін автомобілів за виробником.

- Сторінка з прогнозом розвитку сегмента гібридних автомобілів була реалізована за допомогою класу «Hybrids». Інформація на сайті відображена у вигляді графіків.

Розроблене програмне забезпечення для аналізу перспективності збуду гібридних автомобілів відповідає таким нефункціональним вимогам:

- Система кросбраузерна (сумісна з усіма браузерами, які наведені у підрозділі 1.3. «Загальні вимоги до системи»).
- Система витримує одночасне підключення до серверу 200 користувачами.
- Розроблено зручний для користувача інтерфейс.

4.2. Тестування системи

У широкому сенсі, тестування – це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування і аналіз отриманих результатів.

Тестування потрібно для того, щоб підвищити ймовірність того, що додаток, призначений для тестування, буде працювати правильно при будь-яких обставинах та буде відповідати всім описаним вимогам. Також, надання актуальної інформації про стан продукту на даний момент.

Призначення тестування відрізняються в залежності від етапу створення програмного продукту. На етапі розробки коду програми можна знайти та виправити основні помилки та некоректну поведінку програми, яка попередньо зазначена у функціональних вимогах до ПЗ. На етапі кінцевого тестування потрібно повністю переконатися у стабільній роботі розробленої системи. Під час внесення нових даних або нових

функціональних можливостей можуть виникати незаплановані баги, тому їх слід усунути під час подальшого супроводу.

Дану систему було протестовано за допомогою методу «Чорного ящику». Процес дослідження ПЗ на баги здійснюється шляхом генерації тестових випадків виходячи з аналізу функціональної специфікації (документ, який описує бажані характеристики системи і підсумковий результат) або певних елементів системи.

Переваги «Чорного ящику»:

- Аналіз ПЗ на наявність дефектів здійснюється як би на призначеному для користувача рівні (з позиції користувача) без поглиблення у внутрішню структуру системи.
- Потрібно значно менше часу, завдяки тому, що тест-кейси можна складати відразу після складання специфікації.

Недоліки «Чорного ящику»:

- Пропуск моментів, які не прописані в специфікації, але присутні в функціональності коду.
- Невелике охоплення – тестування піддаються лише кілька вступних значень.
- При нечіткої специфікації можливі труднощі в складанні тестів.

За результатами цього тестування до розроблюваного програмного забезпечення у вигляді вебдодатку були додані повідомлення на клієнтській частині про відповідні помилки, які виникають при вводі некоректних даних.

4.3. Рекомендації щодо подальшого вдосконалення

Вибір функцій розробленого програмного забезпечення вебдодатку є достатньо широким з огляду на зручність використання користувачами та задовольняє багато потреб. Проте, щоб дана система була завжди конкурентоспроможною та універсальною, її потрібно весь час вдосконалювати.

Після аналізу розробленого ПЗ було сформовано ряд додаткових функцій, які б покращили дану систему:

- Порівняльний аналіз гібридних автомобілів та електромобілів.
- Зробити додаткові функціональні можливості для електромобілів, а саме: карта, на якій зображені станції швидкої підзарядки.
- Оновлювати інформацію зміни цін на паливо за кожний минулий місяць.
- Почати співпрацювати із мережами заправок, повідомляти користувачів про існуючі акції та де вигідніше заправитись.

Якщо даний проєкт продовжуватиме активно розвиватися, то у майбутньому можлива перспектива переходу всього населення на більш екологічний вид транспорту.

ВИСНОВКИ

Метою даного дипломного проєкту було розроблення зручного програмного забезпечення для аналізу перспектив збуту гібридних автомобілів з використанням технології Big Data. Таке програмне забезпечення підвищить продуктивність користувачів та знизить витрати часу на аналіз виконаних задач.

Було проведено дослідження предметної галузі та детальний аналіз існуючих рішень. Враховуючи дані, зібрані під час аналізу, було створено та описано функціональні та нефункціональні вимоги, а саме вимоги до безпеки та інтерфейсу.

Для кожного з компонентів розроблюваного веб-сервісу, а саме серверної та клієнтської частин і СКБД було проаналізовано найпопулярніші засоби реалізації. Було обрано платформу C# для програмування серверної частини, фреймворк React для розробки клієнтської частини та Microsoft SQL Server в якості СКБД.

Було створено веб-сервіс з використанням MVC архітектури. Реалізоване програмне забезпечення має:

- Надавати розгорнуту інформацію про ціни на автомобілі та зміни цін на паливо для них у вигляді графіків.
- Надавати порівняльну статистику кількості шкідливих викидів у атмосферу автомобілями в залежності від витрати палива та його виду.
- Надавати інформацію щодо прогнозу розвитку сегмента гібридних автомобілів.
- Підтримувати відповідне шифрування даних задля підтримання інформаційної безпеки.

Розроблене програмне забезпечення виконане у повному обсязі та відповідає всім описаним функціональним вимогам. Проведено тестування

з використанням методу «Чорного ящику», помилки та недоліки, що були виявлені виправлені.

Порівняння веб-сервісу з існуючими аналогами показало, що розроблене програмне забезпечення має переваги над іншими рішеннями, що робить його конкурентоспроможним на ринку.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Big Data [Електронний Ресурс]. Режим доступу: https://www.it.ua/ru/knowledge-base/technology-innovation/big-data-bolshie-dannye.
2. What is MapReduse [Електронний Ресурс]. Режим доступу: https://blog.sqlauthority.com/2013/10/09/big-data-buzz-words-what-is-mapreduce-day-7-of-21/.
3. Time Series Analysis and Its Applications [Текст] / Robert H. Shumway. – 3rd edition. – Springer, 2010. – 596 p.
4. Python documentation [Електронний Ресурс]. Режим доступу: https://docs.python.org/3/.
5. C# in Depth [Текст] / Jon Skeet. – Manning Publications. – 4th edition. – 528 p.
6. ASP.NET Core in Action [Текст] / Andrew Loc. – Shelter Island: Manning, 2018. – 304 p.
7. PostgreSQL documentation [Електронний Ресурс]. Режим доступу: https://www.postgresql.org/docs/.
8. MongoDB documentation [Електронний Ресурс]. Режим доступу: https://docs.mongodb.com/.
9. Programming Asp.Net MVC 4 [Текст] / Jess Chadwick. – O'Reilly Media, 2012. – 1st edition. – 490 p.
10. React [Електронний Ресурс]. Режим доступу: https://reactjs.org/.
11. SQL Server 2019 [Електронний Ресурс]. Режим доступу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019.

ДОДАТКИ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ПЕРСПЕКТИВНОСТІ ЗБУДУ ГІБРИДНИХ АВТОМОБІЛІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ BIG DATA

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Аліна КОРСАКОВА

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data, створено з використанням .NET Core, React.js, Microsoft SQL Server, Scrapy.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок web-ресурсу;

2) функціональна працездатність розробленого API;

3) забезпечення належного рівня безпеки даних;

4) зручність роботи з web-сайтом;

5) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-ресурсу перевіряється шляхом:

1) мануальне (ручне) тестування інтерфейсу;
2) мануальне (ручне) тестування API з використанням Swagger;
3) покриття розроблених модулів функціональними тестами;
4) покриття розроблених модулів модульними тестами;
5) тестування web-ресурсу в різних web-браузерах;
6) тестування при максимальному навантаженні;
7) тестування стабільності роботи при різних умовах;
8) тестування інтерфейсу.

Для написання та прогону модульних та функціональних тестів було використано пакет Jest.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ПЕРСПЕКТИВНОСТІ ЗБУТУ ГІБРИДНИХ АВТОМОБІЛІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ BIG DATA

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Аліна КОРСАКОВА

2020

ЗМІСТ

1. Опис структури програмної платформи

Користувацька частина програмної платформи являє собою веб-додаток, розроблений як SPA (односторінковий додаток). Мова інтерфейсу – англійська. Структура сторінки скаладється з навігаційної панелі, що розташована зверху, та основного тіла. Навігаційна панель містить наступні елементи:

- вкладка «Home», яка веде на головну сторінку;
- вкладка «Hybrids» для відкриття сторінки з прогнозом розвитку сегмента гібридних автомобілів;
- вкладка «CarSales», на якій статистика продажів автомобілів;
- вкладка «FuelSales», на якій статистика зміни цін на автомобільне паливо;
- вкладка «Pollutions», на якій статистика викидів CO_2 автомобілями, в залежності від виду палива;
- вкладка «Forecast», на якій прогноз кількості зареєстрованих гібридних автомобілів в Україні;
- кнопка «Login» для відправки даних для авторизації (якщо користувач не авторизований);
- кнопка «Register» для реєстрації (якщо користувач не авторизований);
- кнопка «Profile» для відкриття сторінки профілю (якщо користувач авторизований);
- кнопка «Log out» для завершення сесії (якщо користувач авторизований).

Навігаційна панель наявна завжди та виділяє ту вкладку, яка відкрита на даний момент.

2. Опис вмісту web-сторінок

На сторінці, що доступна за вкладкою «Home» міститься інформація про програмну платформу, а також інформація про вміст вкладок.

На сторінці, що доступна за вкладкою «Pollutions» (рис. 1.) , на якій буде відображена інформація з порівняльною статистикою кількості шкідливих викидів у атмосферу автомобілями. Також на цій сторінці можна сортувати статистику викидів за видом палива.

«CarSales» – сторінка, на якій буде статистика продажів автомобілів, починаючи з 2016 року. Також на цій сторінці можна сортувати статистику продажів за роком (рис. 2).

На сторінці, що доступна за вкладкою «FuelSales» (рис. 3), на якій буде відображена інформація зміни цін на автомобільне паливо. Також на цій сторінці можна сортувати вивід зміни цін на паливо за роком.

На сторінці, що доступна за вкладкою «Hybrids» (рис. 4), на якій буде відображена прогноз зареєстрованих гібридних автомобілів в Україні.

На сторінці, що доступна за вкладкою «CarPrice» (рис. 5), на якій буде відображені ціни на автомобілі. Також на цій сторінці можна сортувати вивід зміни цін на автомобілі за виробником.

На сторінці «Profile» доступна інформація про користувача та можливість оновлювати свої персональні дані, а саме: ім’я, поштова скринька, логін та пароль.

Може зберігати графіки у форматі .png файлу, а саме: графік порівняльної статистики кількості шкідливих викидів у атмосферу автомобілями, графік цін на автомобілі та зміни цін на пальне, графік прогнозу розвитку сегмента гібридних автомобілів. Користувач має можливість підписки на оновлення статистики, після чого на вказану при реєстрації поштову скриньку буде приходити інформація про оновлення даних на сайті.

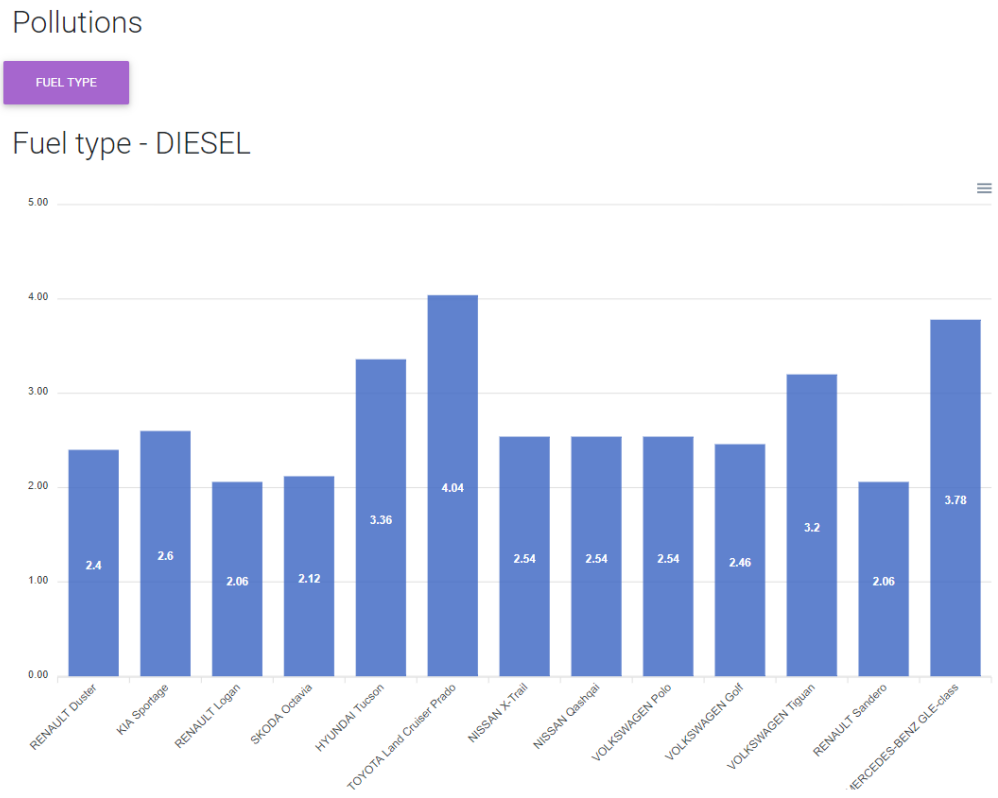


Рис. 1. Сторінка з кількістю викідів шкідливих речовин автомобілями.

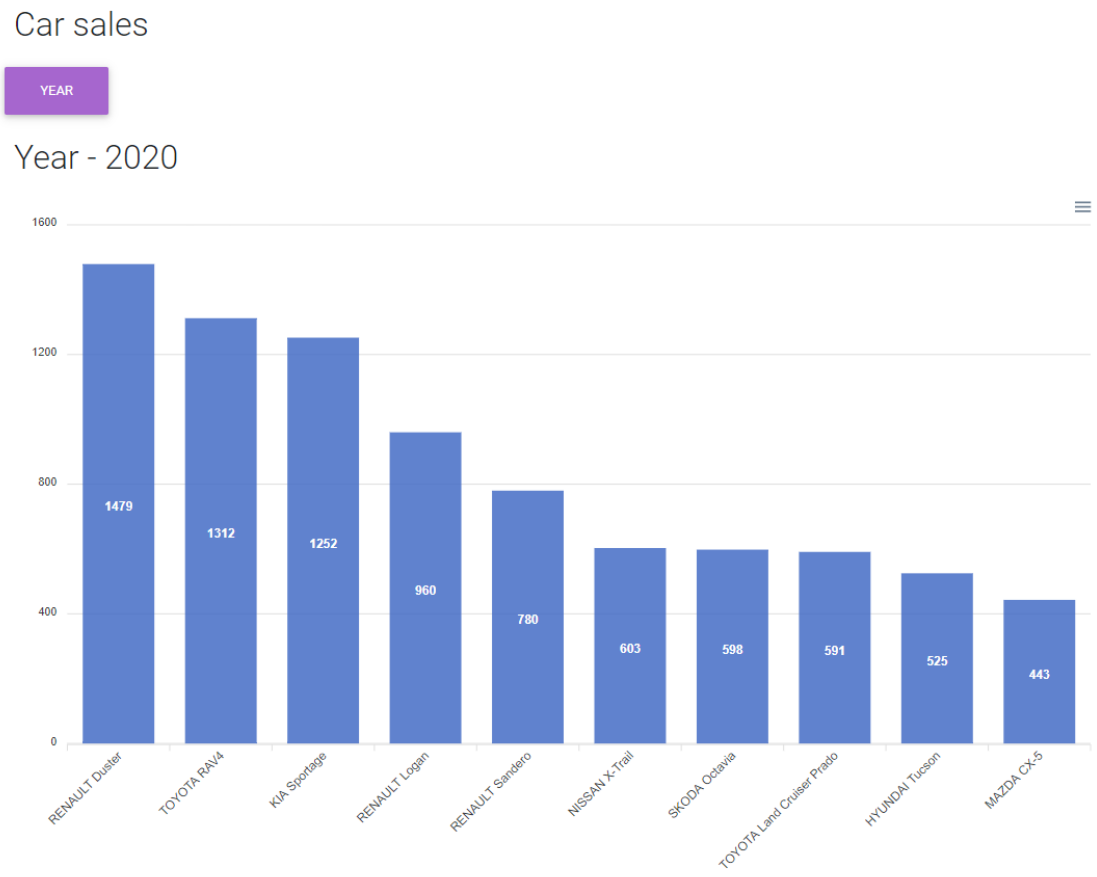


Рис. 2. Сторінка з статистикою продажів автомобілів.

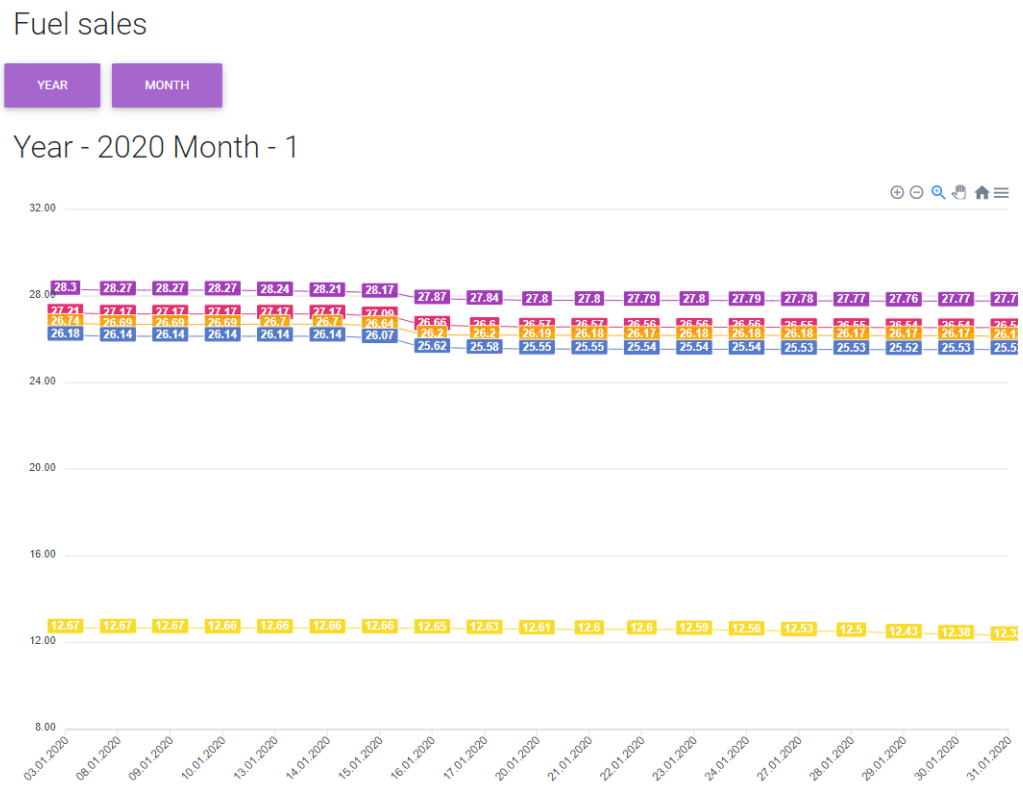


Рис. 3. Сторінка з статистикою зміни цін на автомобільне паливо.

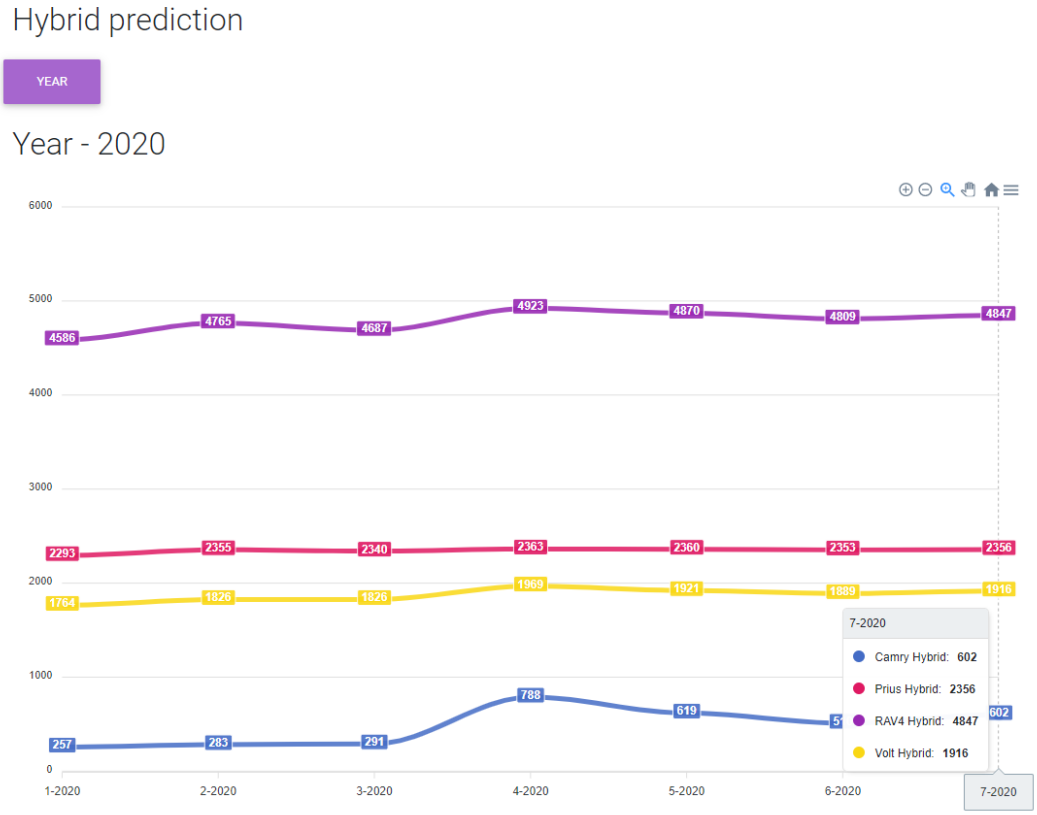


Рис. 4. Сторінка з прогнозом зареєстрованих гібридних автомобілів.

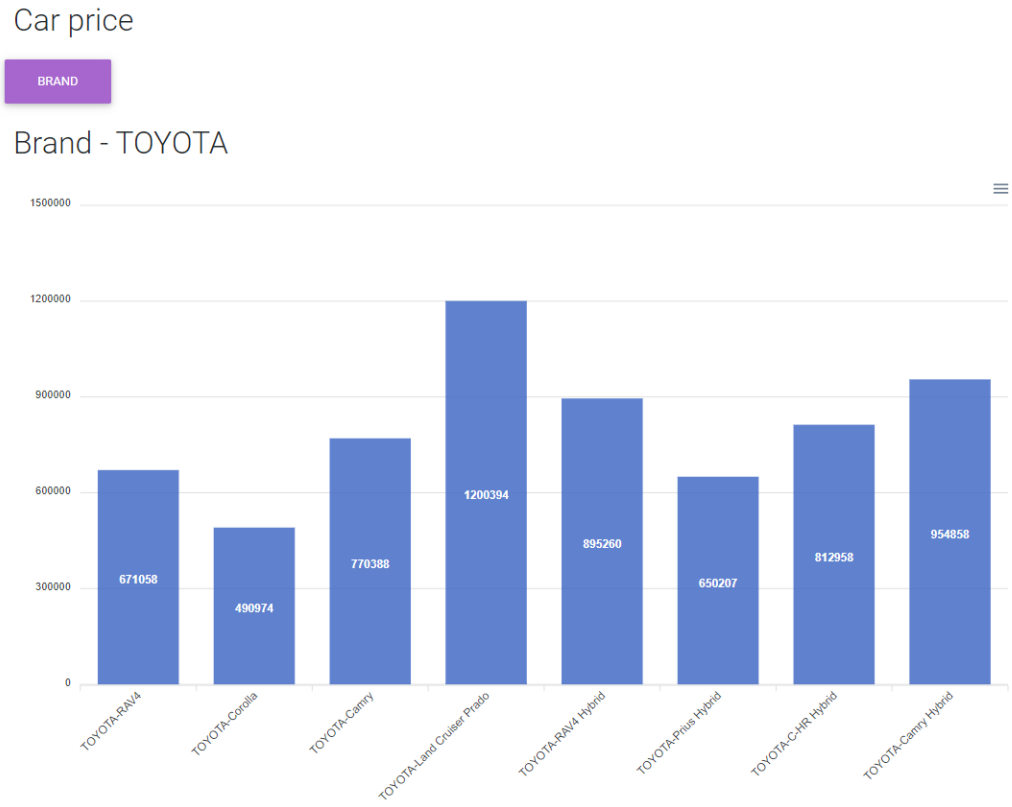


Рис. 5. Сторінка з цінами на автомобілі.

3. Процедура реєстрації користувача

Для реєстрації користувачу потрібно натиснути кнопку «Register», що знаходиться у навігаційному блоці зверху. Після чого буде відкрите модальне вікно, у якому потрібно ввести необхідні дані (рис. 6). Якщо реєстрація пройшла успішно, то користувач зареєстрований у системі, інакше – отримає повідомлення про помилку.

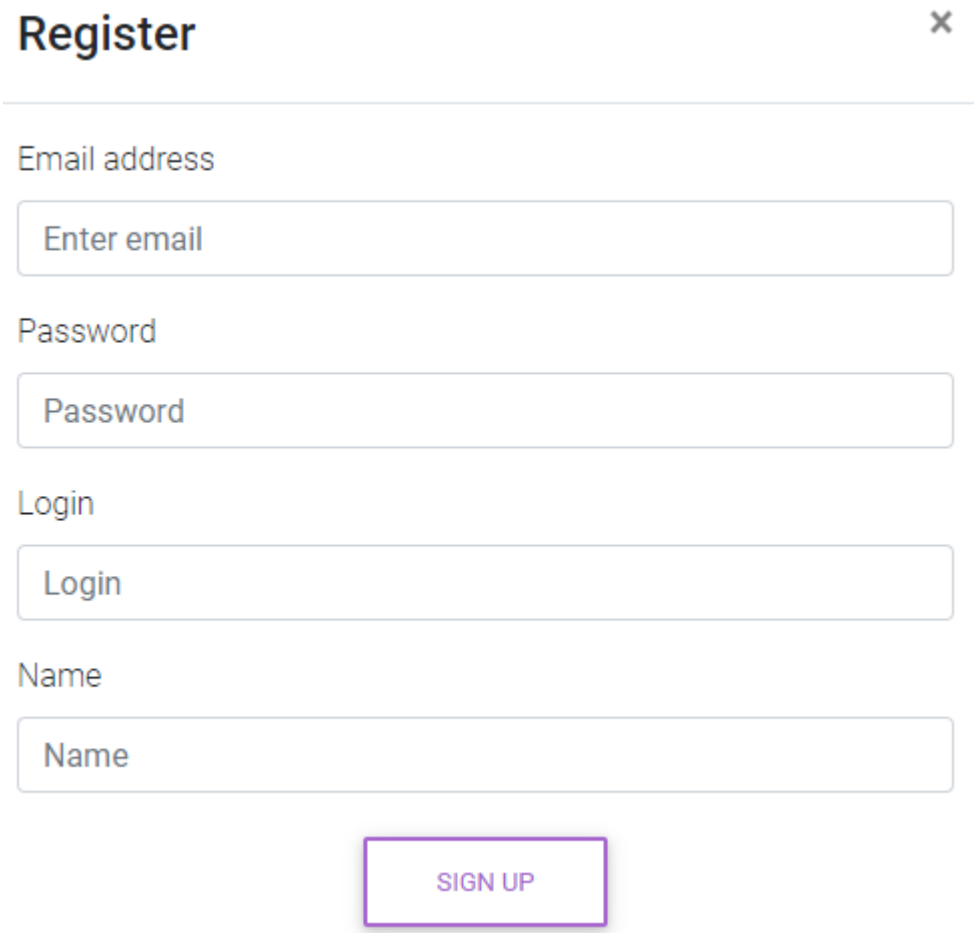


Рис. 6. Форма реєстрації користувача.

4. Процедура авторизації користувача

Для авторизації користувач повинен ввести необхідні дані (логін, який був використаний при реєстрації та пароль) у поля вводу у модальному вікні (рис. 7), після чого натиснути кнопку «Login». Якщо дані вірні – користувача буде авторизовано, інакше – з’явиться повідомлення про помилку.

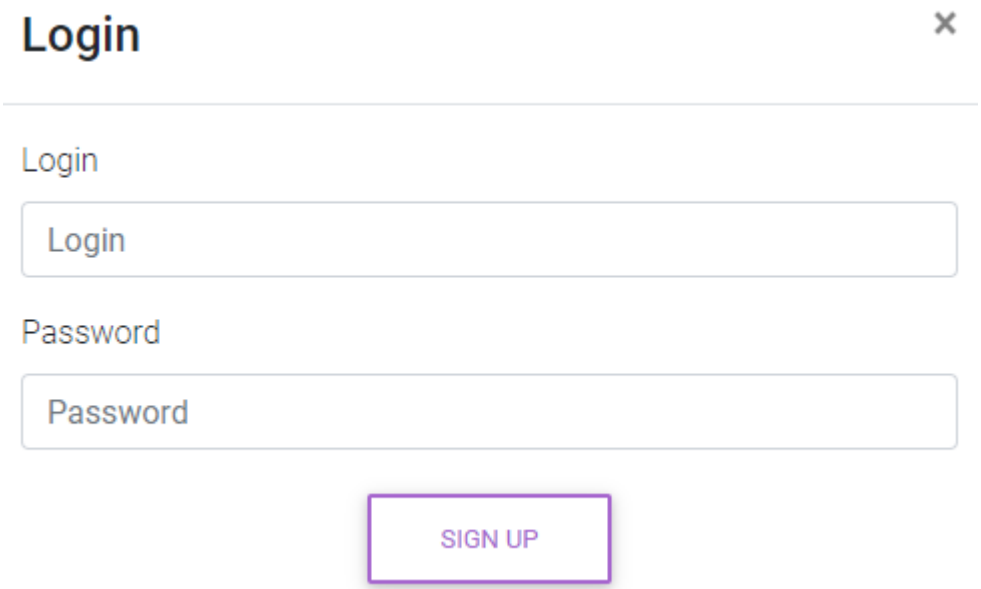
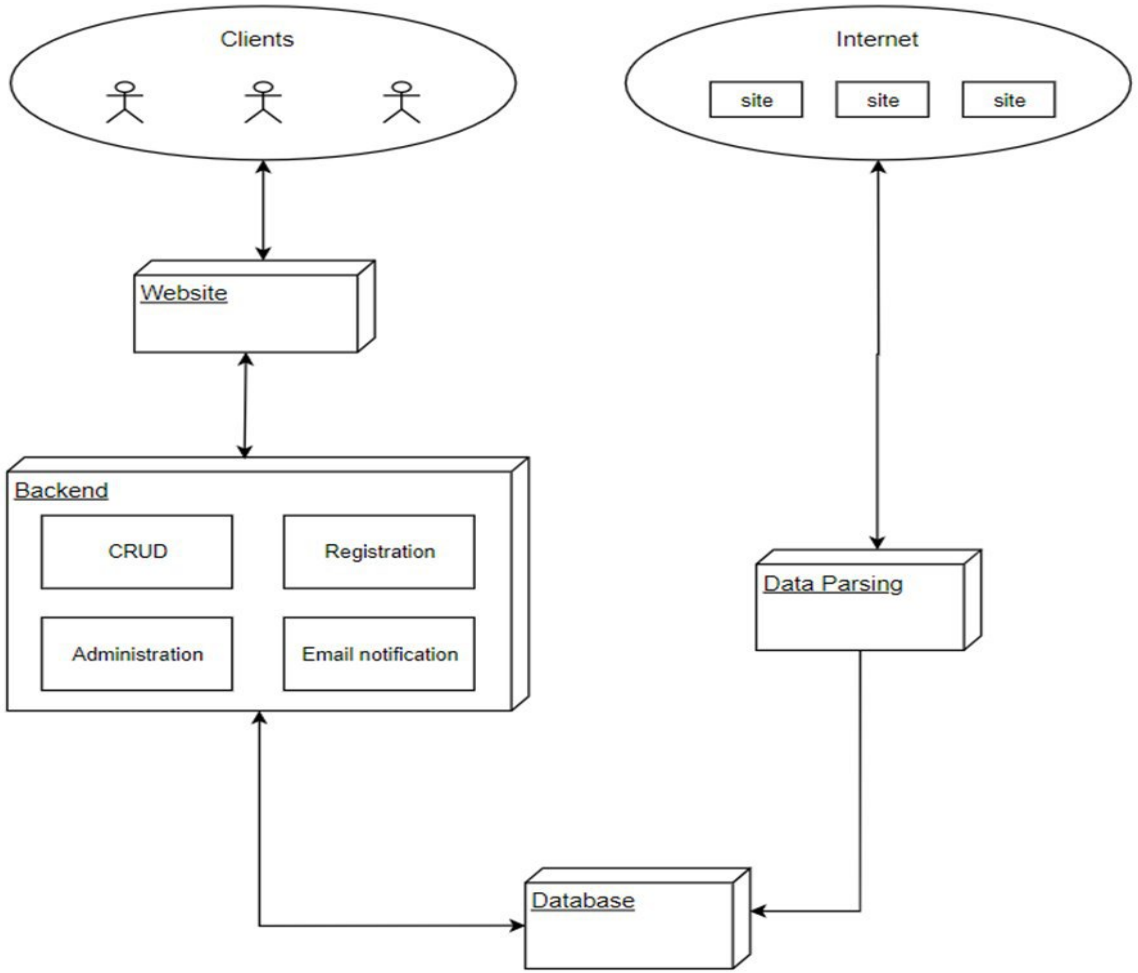


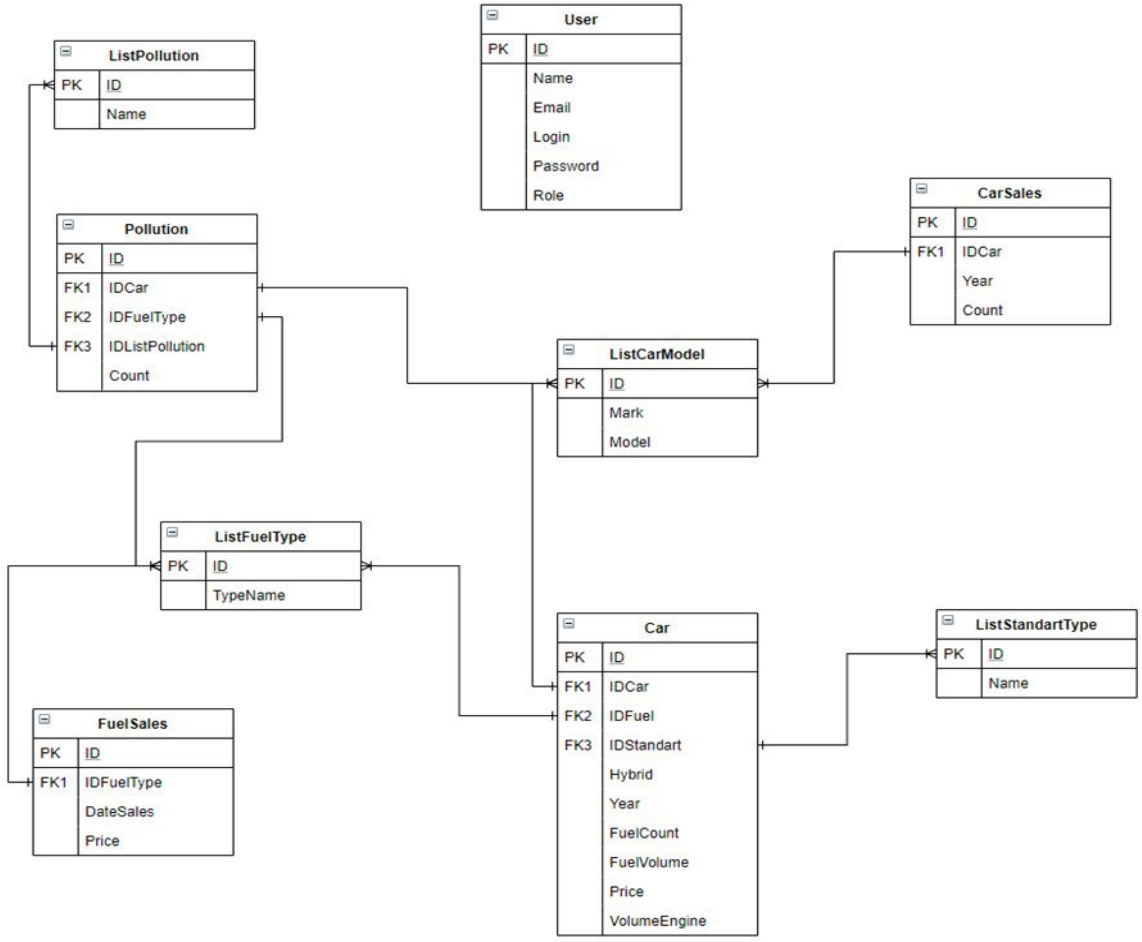
Рис. 7. Форма авторизації користувача.

Додаток 1

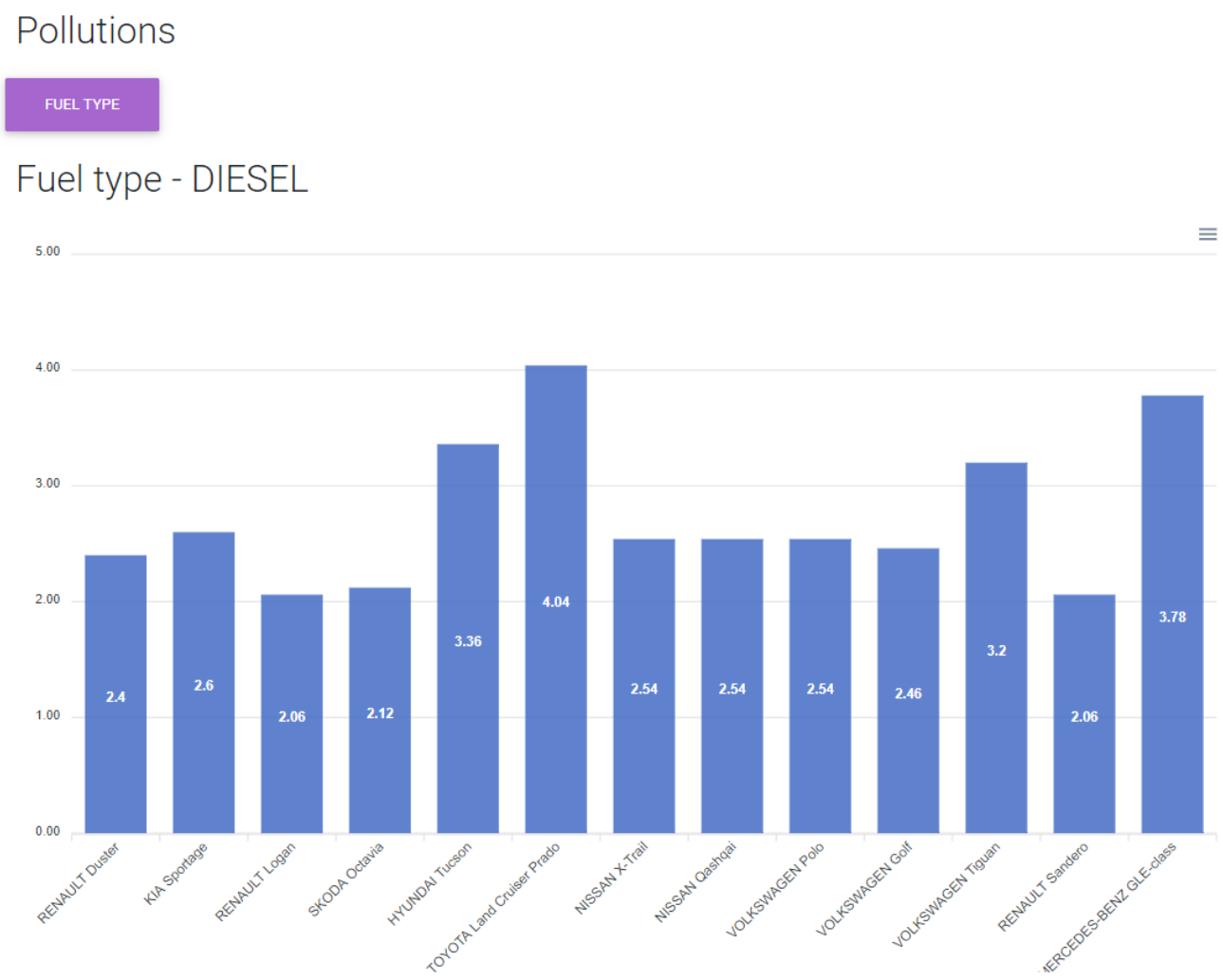
Копії графічних матеріалів



ДП.045440-05-34
Програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data. Діаграма компонентів системи. UML-діаграма компонентів



ДП.045440-06-99
Програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data.База даних. Схема бази даних

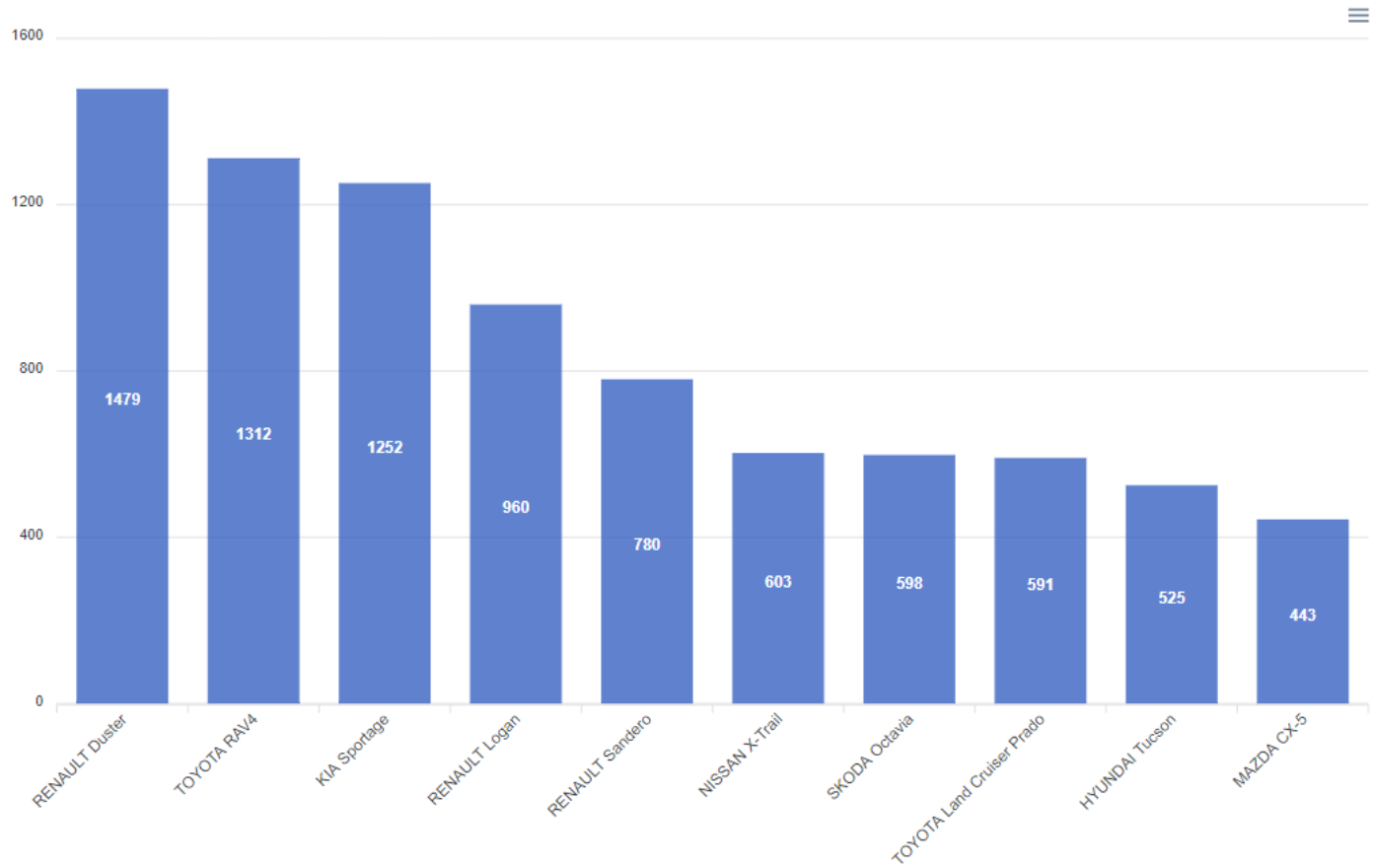


Корсакова Аліна Вадимівна, гр. КП-61

Car sales

YEAR

Year - 2020

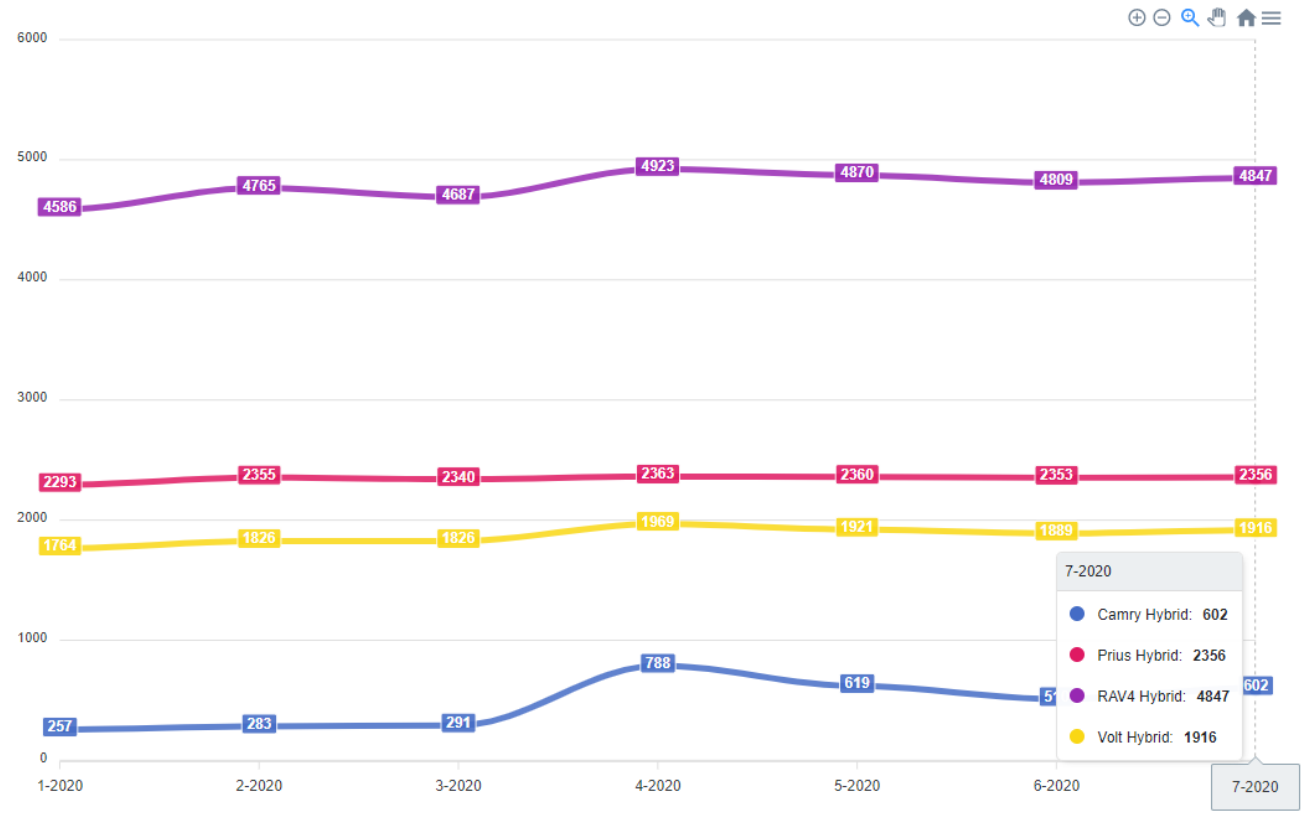


Корсакова Аліна Вадимівна, гр. КП-61

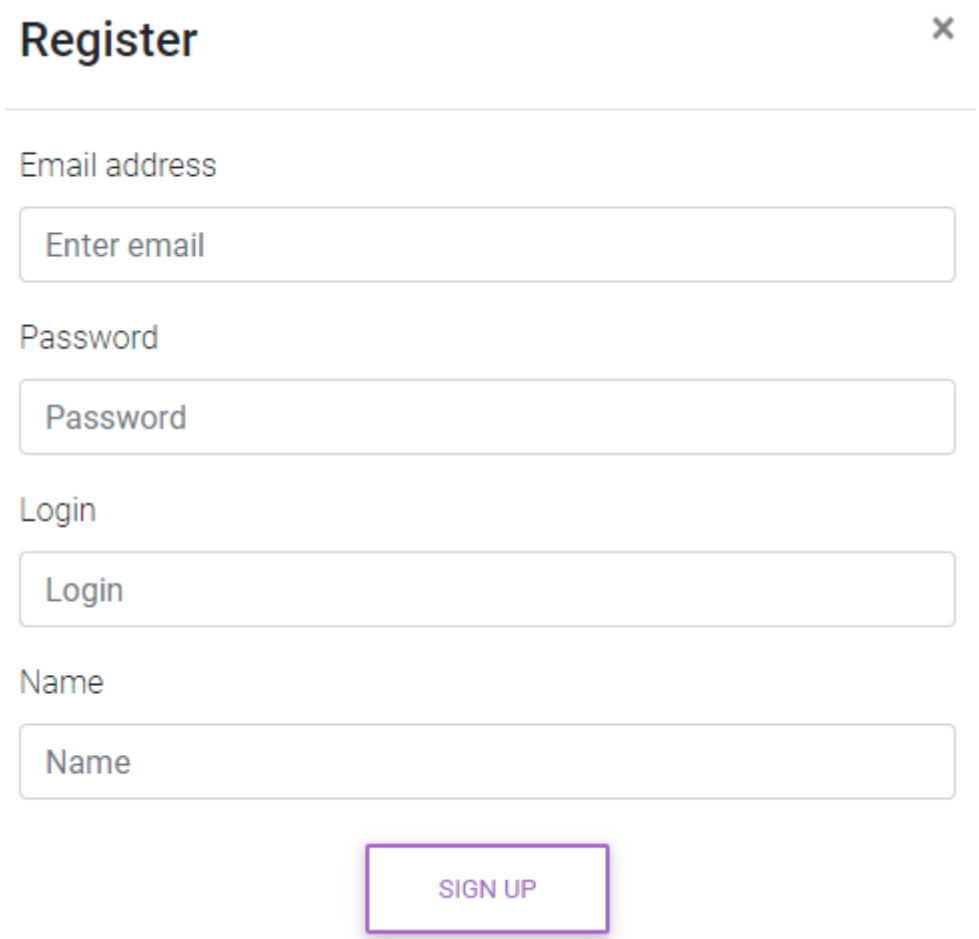
Hybrid prediction

YEAR

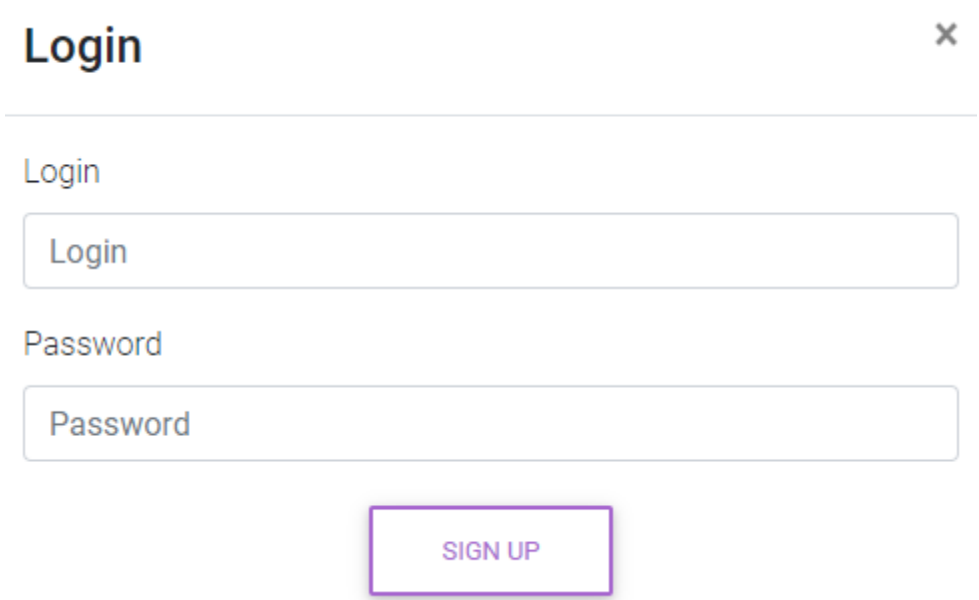
Year - 2020



Корсакова Аліна Вадимівна, гр. КП-61



Корсакова Аліна Вадимівна, гр. КП-61



Корсакова Аліна Вадимівна, гр. КП-61

Додаток 2

Текст програми

```
import React, { Component } from 'react';
import Chart from "react-apexcharts";
import { MDBJumbotron, MDBDropdown, MDBDropdownItem, MDBDropdownMenu,
MDBDropdownToggle, MDBRow } from 'mdbreact';
import axios from 'axios';
const salesBars = {
    options : {
        chart : {
            id : 'basic'
        },
        xaxis : {
            categories : []
        },
        colors:['#3B6BC3', '#E91E63', '#9C27B0']

    },
    series : [
        {
            name : 'sells',
            data : [],
        }
    ]
}
export default class Charts extends Component {
    constructor(props) {
        super(props);
        this.state = {
            selYear: 2020,
            // ...fuelPrices
            ...salesBars
        }
    }

    handleClick = async e => {
        await this.setState({ selYear: parseInt(e.target.innerHTML) })
        this.loadData();
    }

    loadData = async () => {
        try {
            const result = await
axios.get('http://localhost:5000/api/Api/GetCarSales', { params: { year:
this.state.selYear } });

            result.data.data.cars.forEach((element) => {
                this.state.options.xaxis.categories.push(element.Brand + ' '
+ element.Model);
                this.state.series[0].data.push(element.Count);
            })

            console.log(this.state.options.xaxis.categories);
            console.log(this.state.series[0].data);

        } catch (error) {
            console.log(error);
        }
        console.log(this.state)
    }

    render() {
        return (
```

```
            <MDBJumbotron>
                <h2>Car sales</h2>
                <p></p>
                <MDBRow>
                    <MDBDropdown style={{ padding: '0px 0px 15px 0px',
margin: '-35px 0px 0px 0px' }}>
                        <MDBDropdownToggle
variant="link">YEAR</MDBDropdownToggle>
                        <MDBDropdownMenu>
                            <MDBDropdownItem
onClick={this.handleClick}>2016</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2017</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2018</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2019</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2020</MDBDropdownItem>
                        </MDBDropdownMenu>
                    </MDBDropdown>
                </MDBRow>
                <p></p>
                <Chart
                    options={this.state.options}
                    series={this.state.series}
                    type="bar"
                    width="100%"
                />
            </MDBJumbotron>

        );
    }
}


public ResponseCarPrice(string xmlData, string xmlResult)
        {
            XElement XmlResult = XElement.Parse(xmlResult);
            this.ResultCode =
Convert.ToInt32(XmlResult.Element("ResultCode").Value);
            this.ResultType = XmlResult.Element("ResultType").Value;
            this.ResultStr = XmlResult.Element("ResultText").Value;

            Data = null;

            if ((ResultCode == 0) && (!String.IsNullOrEmpty(xmlData)))
            {
                var doc = XDocument.Parse(xmlData);
                var customer = doc.Element("Data");
                try
                {
                    var obj = Helpers.Deserialize<List<CarPrice>>(xmlData);

                    Data = JToken.FromObject(obj);
                }
```

```
                catch (FormatException)
                {
                    Data = null;
                }
            }
        }

    [Produces("application/json")]
    [Consumes("application/json")]
    [Route("api/[controller]")]
    public class ApiController : Controller
    {
        private readonly IOptions<AppSettings> _settings;
        private readonly IServiceTest _service;

        /// <remarks/>
        public   ApiController(IServiceTest   service,   IOptions<AppSettings>
settings)
        {
            _settings = settings;
            _service = service;
        }

        [HttpGet]
        [Route("GetFuelSales")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public async Task<IActionResult> GetFuelSales(int year)
        {
            IResponse rez = await _service.GetFuelSales(year);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }
        [HttpGet]
        [Route("GetPollution")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
```

```
        public async Task<IActionResult> GetPollution()
        {
            IResponse rez = await _service.GetPollution();
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }

        [HttpGet]
        [Route("GetCarSales")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public async Task<IActionResult> GetCarSales(int year)
        {
            IResponse rez = await _service.GetCarSales(year);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }

        [HttpGet]
        [Route("GetCarPrice")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public async Task<IActionResult> GetCarPrice(string Brand)
        {
```

```
            IResponse rez = await _service.GetCarPrice(Brand);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }

        [HttpGet]
        [Route("GetHybridPrediction")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public async Task<IActionResult> GetHybridPrediction(int year)
        {
            IResponse rez = await _service.GetHybridPrediction(year);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);

                Log.Warning(JsonConvert.SerializeObject(rez));
            }
            return response;
        }
        [HttpPost]
        [Route("AccountRegistration")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public   async   Task<IActionResult>   AccountRegistration([FromBody]
Registration req)
        {
```

```
            IResponse rez = await _service.AccountRegistration(req);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }

        [HttpPost]
        [Route("AccountLogin")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public async Task<IActionResult> AccountLogin([FromBody] Login req)
        {
            IResponse rez = await _service.AccountLogin(req);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }
        [HttpPost]
        [Route("AccountRecoverPwd")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public    async    Task<IActionResult>    AccountRecoverPwd([FromBody]
RecoverPassword req)
        {
            IResponse rez = await _service.AccountRecoverPwd(req);
            IActionResult response;
```

```
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }
        [HttpPost]
        [Route("AccountChangePwd")]
        [ProducesResponseType(typeof(IResponse), (int)HttpStatusCode.OK)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.BadRequest)]
        [ProducesResponseType(typeof(IResponse),
(int)HttpStatusCode.InternalServerError)]
        public    async    Task<IActionResult>    AccountChangePwd([FromBody]
ChangePassword req)
        {
            IResponse rez = await _service.AccountChangePwd(req);
            IActionResult response;
            if (rez.ResultCode == (int)HttpStatusCode.InternalServerError)
            {
                response = Helpers.WebResponse(
                    rez.ResultCode,
                    rez.ResultType,
                    rez.ResultStr,
                    _settings.Value.MediaTypeStr);
            }
            else
            {
                response                                                    =
Helpers.WebResponseOK(_settings.Value.MediaTypeStr, rez);
            }
            return response;
        }
    }
public class Program
    {
        /// <remarks/>
        public static void Main(string[] args)
        {
            Log.Logger = new LoggerConfiguration()

.MinimumLevel.Fatal().MinimumLevel.Error().MinimumLevel.Warning()
                .WriteTo
                .Console(outputTemplate: "{Timestamp: yyyy-MM-dd HH:mm:ss.fff
zzz} [{Level:u3}] {Message:lj}{NewLine}{Exception}")
                .CreateLogger();

            try
```

```
            {
                Log.Information("Starting webhost...");
                CreateWebHostBuilder(args).Build().Run();
            }
            catch (Exception ex)
            {
                Log.Fatal(ex, "Host terminated unexpectedly");
            }
            finally
            {
                Log.CloseAndFlush();
            }
        }

        /// <remarks/>
        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                //.UseUrls("http://*:5000")
                .UseSerilog()
                .UseStartup<Startup>();
    }


const salesBars = {
    options: {
        chart: {
            type: 'line'
        },
        colors: ['#3B6BC3', '#E91E63', '#9C27B0'],
        dataLabels: {
            enabled: true,
        },
        stroke: {
            curve: 'smooth'
        },
        markers: {
            size: 1
        },
        xaxis: {
            categories: []
        },
        legend: {
            position: 'top',
            horizontalAlign: 'right',
            floating: true,
            offsetY: -25,
            offsetX: -5
        }
    },
    series: [{
        name: 'Name',
        data: []
    }]
}

export default class Charts extends Component {
    constructor(props) {
        super(props);
        this.state = {
            selYear: 2020,
            ...salesBars
        }
    }
```

```
    handleClick = async e => {
        await this.setState({ selYear: parseInt(e.target.innerHTML) })
        this.loadData();
    }

    loadData = async () => {
        try {
            const result = await axios.get(global.config.url +
'/api/Api/GetHybridPrediction', { params: { year: this.state.selYear } });

            const newSeries = [];
            let newOptions;
            const arrCategories = [];

            result.data.data.monthYear.forEach((element) => {
                arrCategories.push(element);
            })

            result.data.data.predictionCars.forEach((element) => {
                let arrDataLine = [];

                element.counts.forEach((item) => {
                    arrDataLine.push(item);
                })

                newSeries.push({ name: element.carName, data: arrDataLine });
            })

            newOptions = {
                chart: {
                    type: 'line'
                },
                colors: ['#3B6BC3', '#E91E63', '#9C27B0', '#FDD835',
'#FFA41B'],
                dataLabels: {
                    enabled: true,
                },
                stroke: {
                    curve: 'smooth'
                },
                markers: {
                    size: 1
                },
                xaxis: {
                    categories: arrCategories
                },
                legend: {
                    position: 'top',
                    horizontalAlign: 'right',
                    floating: true,
                    offsetY: -25,
                    offsetX: -5
                }
            }

            await this.setState({ series: newSeries });
            await this.setState({ options: newOptions });

        } catch (error) {
            console.log(error);
        }
    }
```

```
    render() {
        return (

            <MDBJumbotron>
                <h2>Hybrid prediction</h2>
                <p></p>
                <MDBRow>
                    <MDBDropdown style={{ padding: '0px 0px 15px 0px',
margin: '-35px 0px 0px 0px' }}>
                        <MDBDropdownToggle
variant="link">YEAR</MDBDropdownToggle>
                        <MDBDropdownMenu>
                            <MDBDropdownItem
onClick={this.handleClick}>2019</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2020</MDBDropdownItem>
                        </MDBDropdownMenu>
                    </MDBDropdown>
                </MDBRow>
                <p></p>
                <h2>Year - {this.state.selYear}</h2>
                <p></p>
                <Chart
                    options={this.state.options}
                    series={this.state.series}
                    type="line"
                    width="100%"
                />
            </MDBJumbotron>

        );
    }
}

const salesBars = {
    options: {
        chart: {
            type: 'line'
        },
        colors: ['#3B6BC3', '#E91E63', '#9C27B0'],
        dataLabels: {
            enabled: true,
        },
        stroke: {
            curve: 'smooth'
        },
        markers: {
            size: 1
        },
        xaxis: {
            categories: []
        },
        legend: {
            position: 'top',
            horizontalAlign: 'right',
            floating: true,
            offsetY: -25,
            offsetX: -5
        }
    },
    series: [{
        name: 'Name',
        data: []
    }]
```

```
}

export default class Charts extends Component {
    constructor(props) {
        super(props);
        this.state = {
            selYear: 2020,
            selMonth: 1,
            ...salesBars
        }
    }

    handleClick = async e => {
        await this.setState({ selYear: parseInt(e.target.innerHTML) })
        this.loadData();
    }

    handleMonthClick = async e => {
        await this.setState({ selMonth: parseInt(e.target.innerHTML) })
        this.loadData();
    }

    loadData = async () => {
        try {
            const result = await axios.get(global.config.url +
'/api/Api/GetFuelSales', { params: { year: this.state.selYear, month:
this.state.selMonth } });

            const newSeries = [];
            let newOptions;
            const arrCategories = [];

            result.data.data.DateSaleTxt.forEach((element) => {
                arrCategories.push(element);
            })

            result.data.data.fuelSales.forEach((element) => {
                let arrDataLine = [];

                element.Price.forEach((item) => {
                    arrDataLine.push(item);
                })

                newSeries.push({ name: element.TypeName, data: arrDataLine
});
            })

            newOptions = {
                chart: {
                    type: 'line'
                },
                colors: ['#3B6BC3', '#E91E63', '#9C27B0', '#FDD835',
'#FFA41B'],
                dataLabels: {
                    enabled: true,
                },
                stroke: {
                    //curve: 'smooth',
                    width: 1
                },
                markers: {
                    size: 0,
                    colors: undefined,
                    strokeColors: '#fff',
```

```
                    strokeWidth: 2,
                    strokeOpacity: 0.9,
                    strokeDashArray: 0,
                    fillOpacity: 1,
                    discrete: [],
                    shape: "circle",
                    radius: 2,
                    offsetX: 0,
                    offsetY: 0,
                    onClick: undefined,
                    onDblClick: undefined,
                    showNullDataPoints: true,
                    hover: {
                        size: undefined,
                        sizeOffset: 3
                    }
                },
                xaxis: {
                    categories: arrCategories
                },
                legend: {
                    position: 'top',
                    horizontalAlign: 'right',
                    floating: true,
                    offsetY: -25,
                    offsetX: -5
                }
            }

            await this.setState({ series: newSeries });
            await this.setState({ options: newOptions });

        } catch (error) {
            console.log(error);
        }
    }

    render() {
        return (

            <MDBJumbotron>
                <h2>Fuel sales</h2>
                <p></p>
                <MDBRow>
                    <MDBDropdown style={{ padding: '0px 0px 15px 0px',
margin: '-35px 0px 0px 0px' }}>
                        <MDBDropdownToggle
variant="link">YEAR</MDBDropdownToggle>
                        <MDBDropdownMenu>
                            <MDBDropdownItem
onClick={this.handleClick}>2015</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2016</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2017</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2018</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2019</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleClick}>2020</MDBDropdownItem>
                        </MDBDropdownMenu>
                    </MDBDropdown>
```

```
                    <MDBDropdown style={{ padding: '0px 0px 15px 0px',
margin: '-35px 0px 0px 0px' }}>
                        <MDBDropdownToggle
variant="link">MONTH</MDBDropdownToggle>
                        <MDBDropdownMenu>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>1</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>2</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>3</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>4</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>5</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>6</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>7</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>8</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>9</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>10</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>11</MDBDropdownItem>
                            <MDBDropdownItem
onClick={this.handleMonthClick}>12</MDBDropdownItem>
                        </MDBDropdownMenu>
                    </MDBDropdown>
                </MDBRow>
                <p></p>
                <h2>Year - {this.state.selYear}  Month -
{this.state.selMonth}</h2>
                <p></p>
                <Chart
                    options={this.state.options}
                    series={this.state.series}
                    type="line"
                    width="100%"
                />
            </MDBJumbotron>

        );
    }
}
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ПЕРСПЕКТИВНОСТІ ЗБУТУ ГІБРИДНИХ АВТОМОБІЛІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ BIG DATA

Виконала: Корсакова Аліна Вадимівна

Керівник: Старший викладач кафедри ПЗКС, к.ф.-м.н., Гречко А.В.

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити програмне забезпечення для аналізу перспективності збуту гібридних автомобілів з використанням технології Big Data.

Завдання:

1. Проаналізувати існуючі застосунки.
2. Розробити ПЗ для аналізу перспективності збуту гібридних автомобілів.
3. Протестувати розроблене ПЗ.

АКТУАЛЬНІСТЬ

Аналіз перспектив збуту гібридних автомобілів в Україні на сьогодні є дуже актуальним питанням, яке цікавить як власників авто, так і потенційних ринкових покупців. Використання технології Big Data дозволяє провести якісний і повноцінний аналіз великого обсягу інформації для прийняття правильного рішення в сторону покупки гібридного автомобіля.

АНАЛОГИ СИСТЕМИ

Сайт Міністерства фінансів України

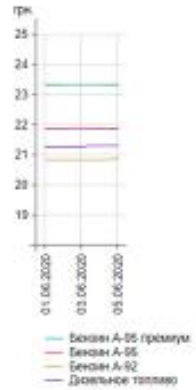
Цены на бензин, дизтопливо, газ на АЗС Украины

последнее обновление: 05.06.2020 14:20

Средние цены на горючее по Украине в июне 2020 (грн./литр)

Дата	А 95+	А 95	А 92	ДТ	Газ
01.06.2020 Пн	23,34	21,86	20,83	21,28	10,38
02.06.2020 Вт	23,34	21,86	20,83	21,28	10,44
03.06.2020 Ср	23,33	21,85	20,82	21,28	10,49
04.06.2020 Чт	23,34	21,86	20,83	21,29	10,53
05.06.2020 Пт	23,34	21,88	20,85	21,31	10,56

* без учета оккупированных территорий (Крыма, Севастополя, части Донбасса)

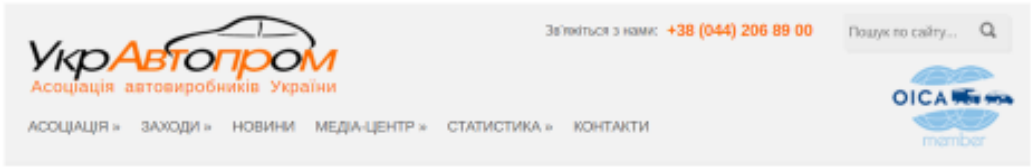


Цены на горючее по годам

- 2015
- 2016
- 2017
- 2018
- 2019
- 2020
  - январь
  - февраль
  - март
  - апрель
  - май
  - июнь

АНАЛОГИ СИСТЕМИ

Сайт УкрАвтопрому



Надії не виправдалися – підсумки року ринку нових легкових авто

Лідери ринку 2018 року за моделями:

RENAULT Duster – 3465 шт.;

KIA Sportage – 3128 шт.;

TOYOTA RAV4 – 2653 шт.;

TOYOTA Camry – 2404 шт.;

RENAULT Logan – 2385 шт.;

HYUNDAI Tucson – 2229 шт.;

NISSAN Qashqai – 2197 шт.;

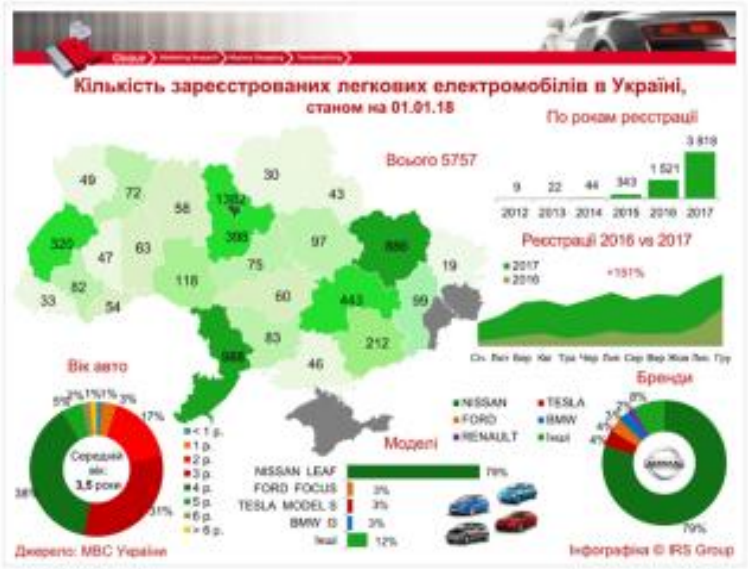
SKODA Octavia – 2138 шт.;

VOLKSWAGEN Polo – 1812 шт.;

NISSAN X-Trail – 1732 шт.

АНАЛОГИ СИСТЕМИ

Сайт IRS Group



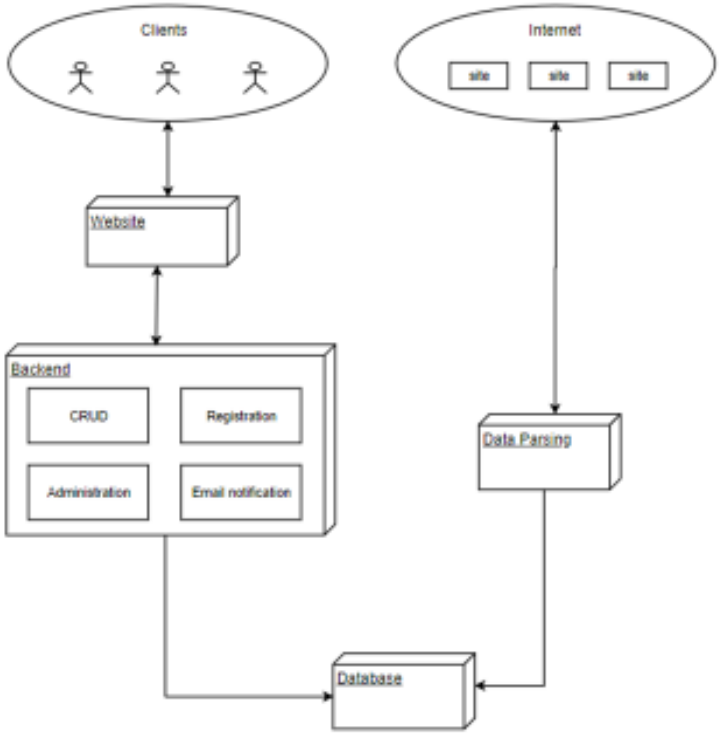
Доступний повний звіт "Аналіз реєстрацій електромобілів в Україні" (близько 60 слайдів).

АРХІТЕКТУРА ПЗ

Дипломну роботу можна розділити на 3 важливих етапи:

- парсинг даних з вебсторінок та наповнення бази даних;
- аналіз отриманих даних;
- відображення результатів аналізу на вебсторінках.

АРХІТЕКТУРА СИСТЕМИ



ЗАСОБИ РЕАЛІЗАЦІЇ

Парсинг даних з вебсторінок та наповнення БД

Scrapy

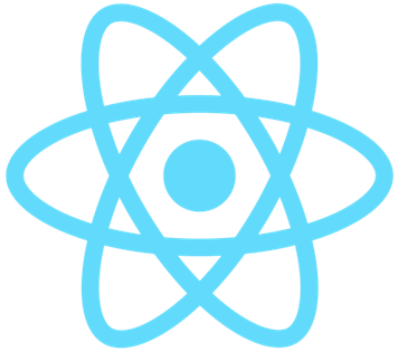
ЗАСОБИ РЕАЛІЗАЦІЇ

Серверна частина

.NET Core

ЗАСОБИ РЕАЛІЗАЦІЇ

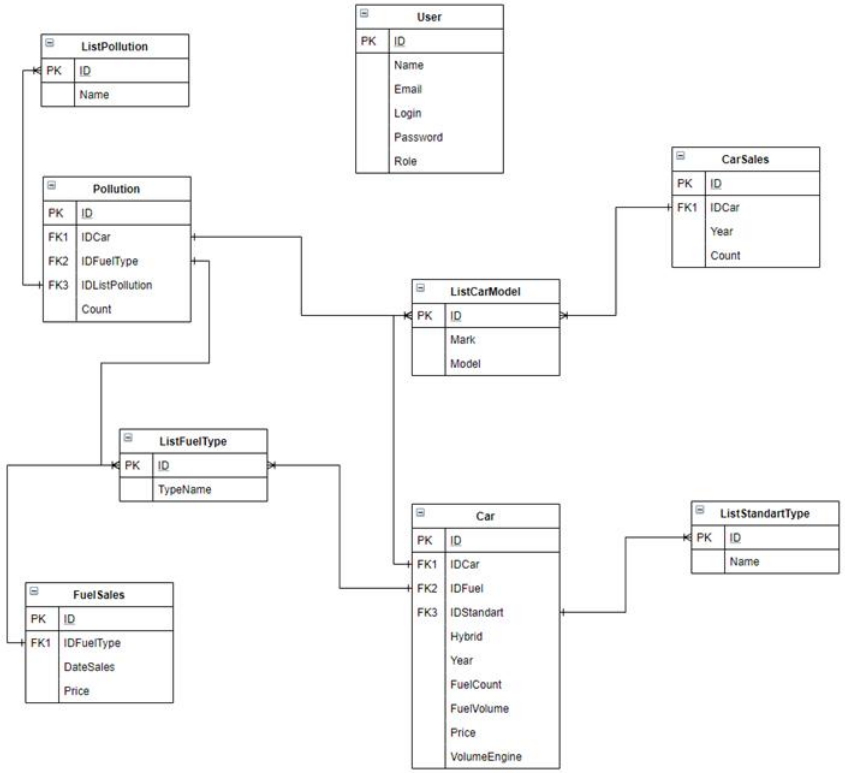
Клієнтська частина



ЗАСОБИ РЕАЛІЗАЦІЇ

База даних

СХЕМА БАЗИ ДАНИХ



РЕЗУЛЬТАТИ РОЗРОБЛЕННЯ СИСТЕМИ

Реалізовані функціональні можливості

Відображення статистичних даних у вигляді таблиці та графіків:

- кількості шкідливих викидів у атмосферу автомобілями;
- зміни цін на автомобільне паливо за різні проміжки часу, починаючи з червня 2015 року;
- прогнозу розвитку сегмента гібридних автомобілів на основі зареєстрованих автомобілів за попередній рік.

РЕЗУЛЬТАТИ РОЗРОБЛЕННЯ СИСТЕМИ

Реалізовані функціональні можливості

Відображення статистичних даних у вигляді таблиці та графіків:

- статистика продажу автомобілів, починаючи з 2016 року;
- діаграма цін на автомобілі за 2020 рік.

РЕЗУЛЬТАТИ РОЗРОБЛЕННЯ СИСТЕМИ

Реалізовані функціональні можливості

Збереження графіків у форматі .png.

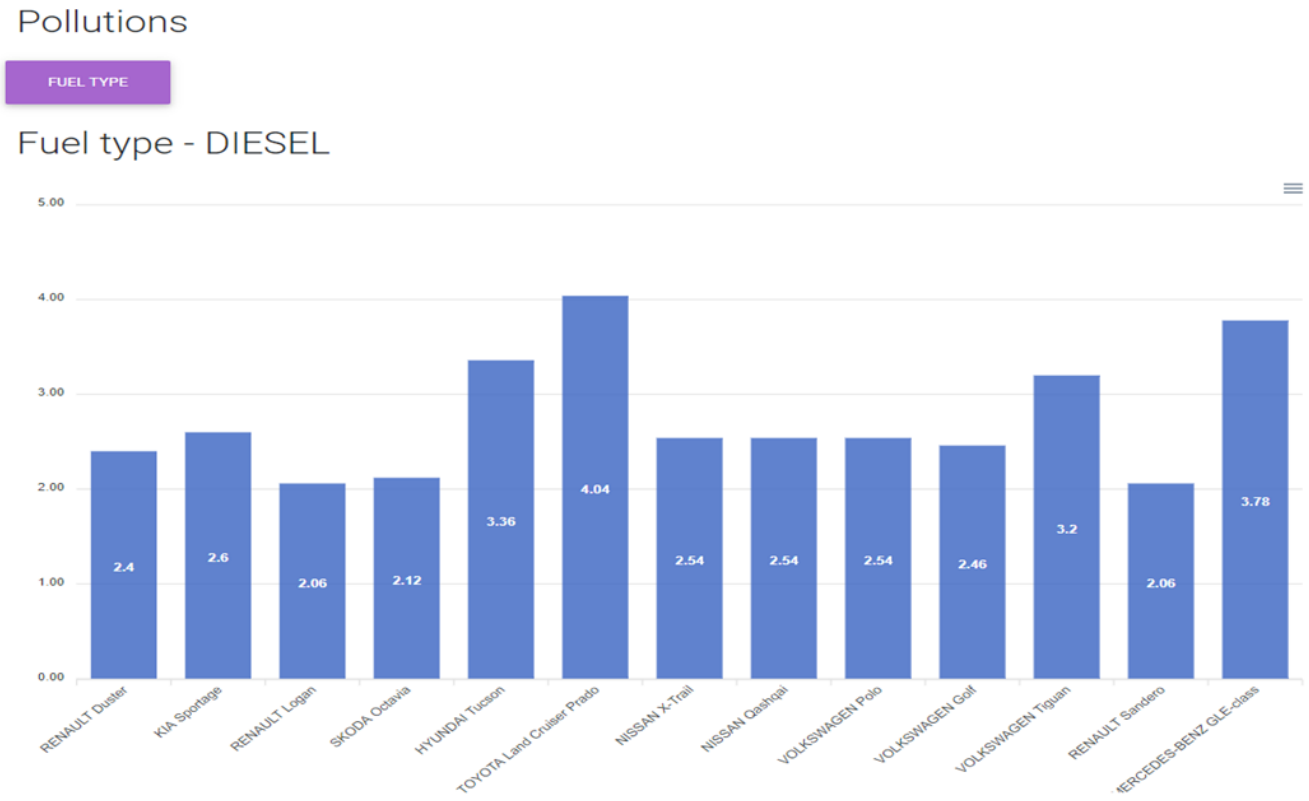
Сортування виводу змін цін на паливо за роком.

Сортування виводу цін на автомобілі за виробником.

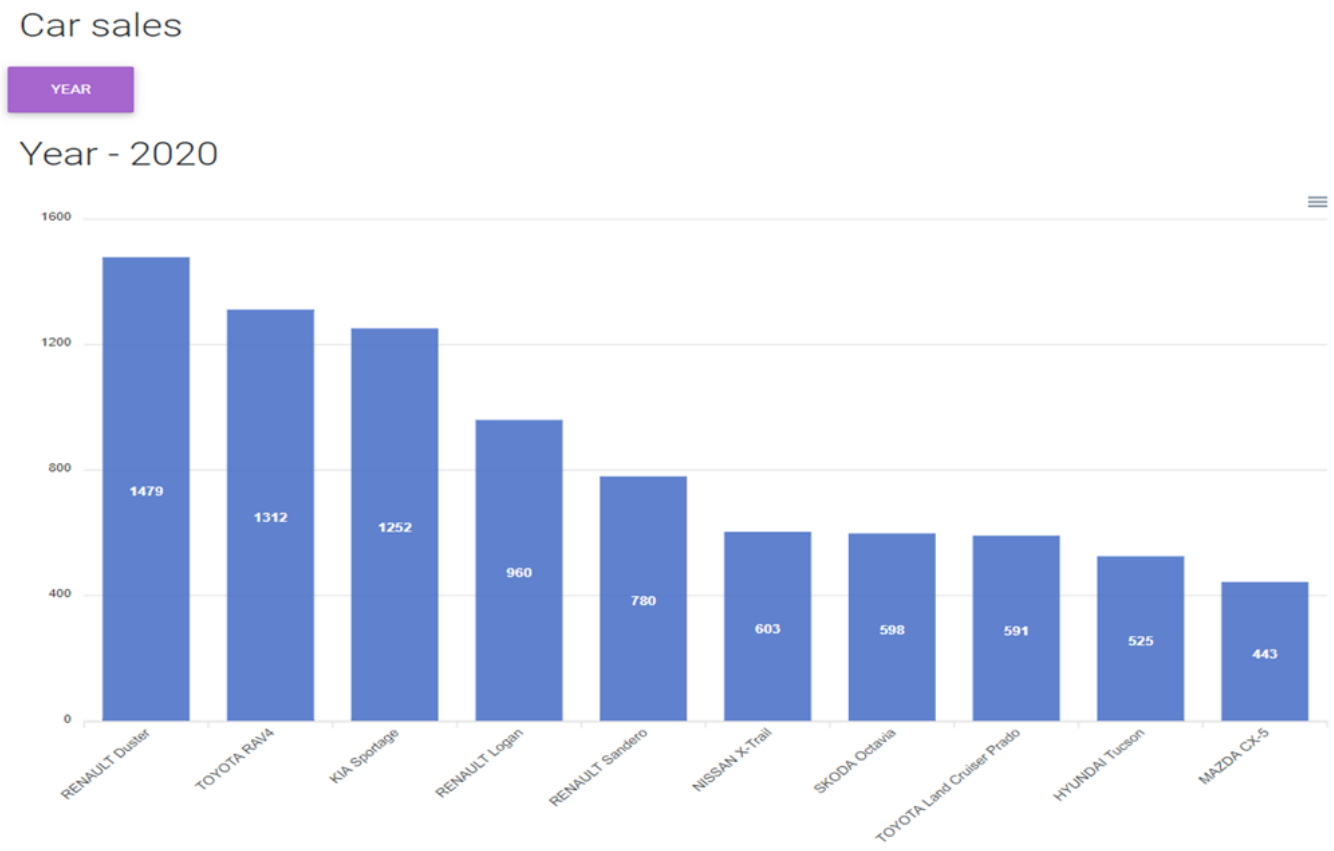
Сортування кількості шкідливих викидів за видом палива.

Сортування статистики продажів автомобілів за роком.

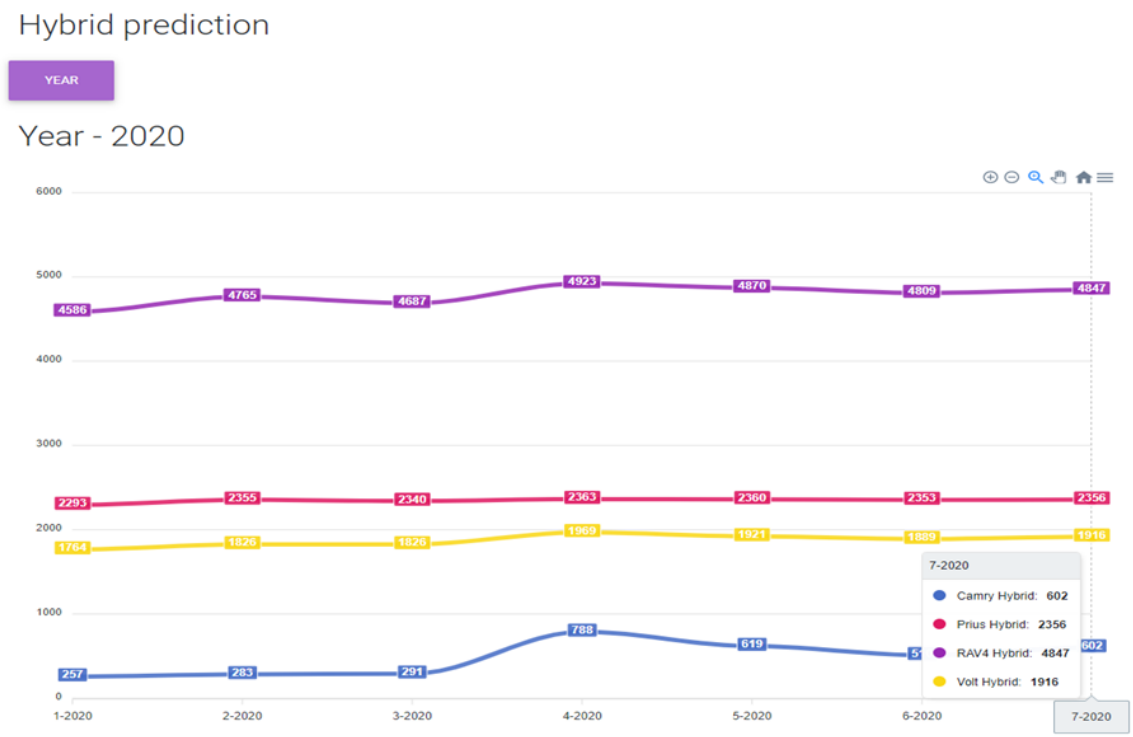
СКРІНШОТ ІНТЕРФЕЙСУ



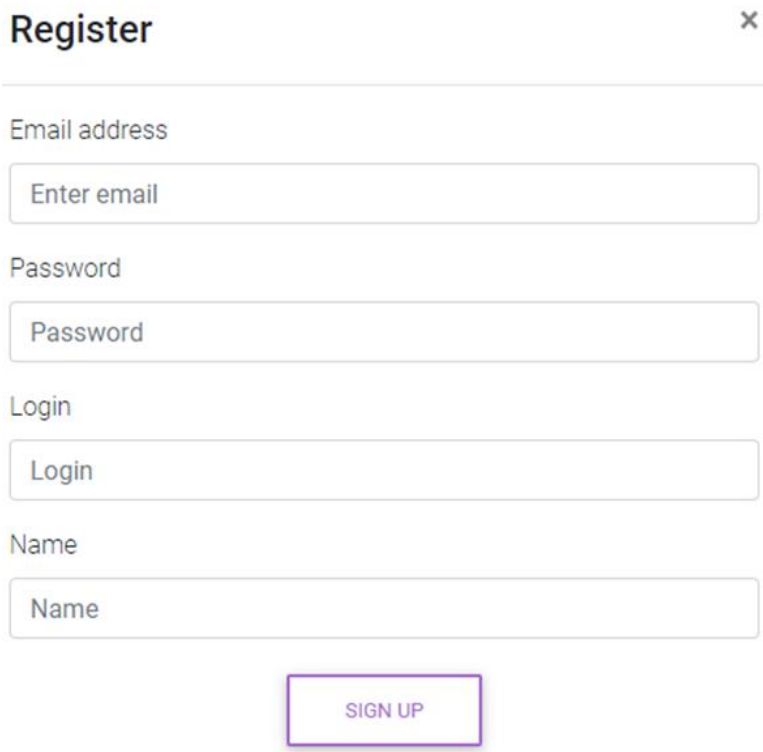
СКРІНШОТ ІНТЕРФЕЙСУ

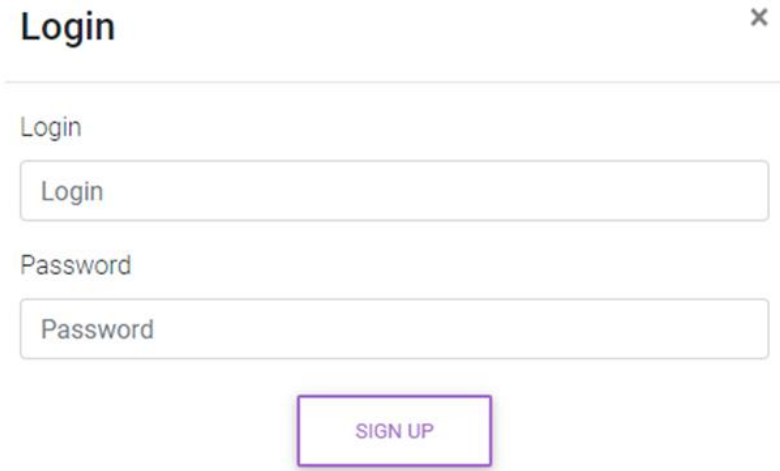


СКРІНШОТ ІНТЕРФЕЙСУ



СКРІНШОТ ІНТЕРФЕЙСУ





ТЕСТУВАННЯ ПЗ

Дану систему було протестовано за допомогою методу «Чорного ящику».

За результатами цього тестування до розроблюваного програмного забезпечення у вигляді вебдодатку були додані повідомлення на клієнтській частині про відповідні помилки, які виникають при вводі некоректних даних.

РЕКОМЕНДАЦІЇ ДЛЯ ПОДАЛЬШОГО ВДОСКОНАЛЕННЯ

- Порівняльний аналіз гібридних автомобілів та електромобілів.
- Зробити додаткові функціональні можливості для електромобілів, а саме: карта, на якій зображені станції швидкої підзарядки.
- Оновлювати інформацію зміни цін на паливо за кожний минулий місяць.
- Почати співпрацювати із мережами заправок, повідомляти користувачів про існуючі акції та де вигідніше заправитись.

ВИСНОВКИ

1. Проаналізовано аналогічні застосунки.

2. Розроблено програмне забезпечення, яке задовольняє функціональним вимогам.

3. Визначені рекомендації щодо подальшого вдосконалення

Дякую за увагу!